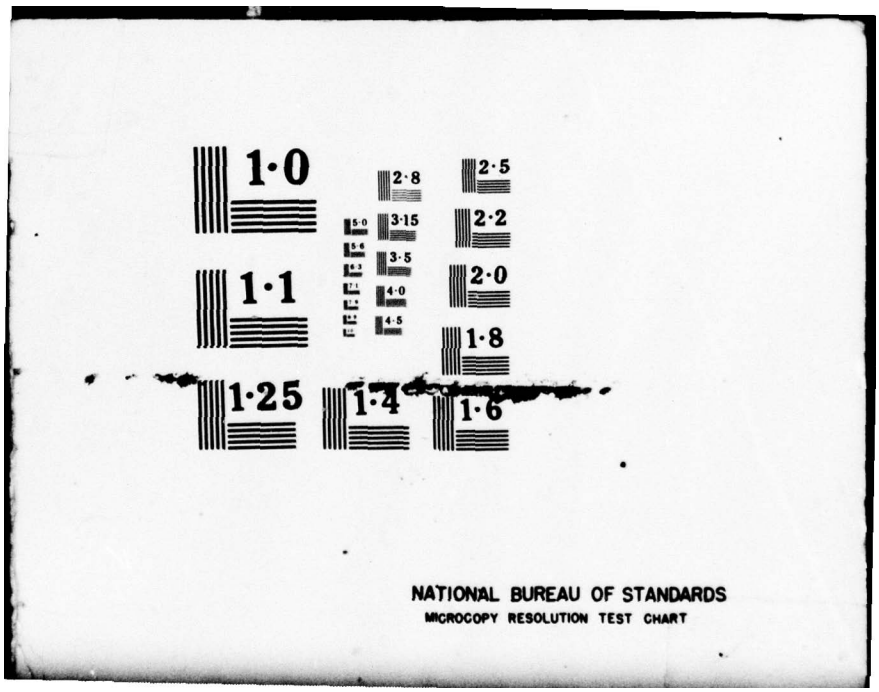
AD-A049 192 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 17/2
THE APPLICATION OF MULTIPLE PROCESSOR COMPUTER SYSTEMS TO DIGIT--ETC(U)
JUN 76 M BARBACCI, L CHANG, S FULLER, A INGLE DCA100-75-C-0066

UNCLASSIFIED SBIE-AD-E100 014 NL

1 OF 2
AD
A049192


1·0
1·1
1·25
2·8
3·15
3·5
4·0
4·5
5·0
1·4
2·5
2·2
2·0
1·8
1·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

The Application of Multiple Processor Computer Systems to Digital Communication Networks.

Mario Barbacci
Lih Chang
Samuel Fuller
Ashok Ingle
Navindra Jain
John Oakley
Daniel Siewiorek

Department of Computer Science
Carnegie-Mellon University
Pittsburgh Pa 15213

This work was supported by the Defense Communications Agency, Defense Communications Engineering Center, under the contract number (DCA100-75-C-0066).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (18) SBIE (19) AD-E100 014

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(6) The Application of Multiple Processor Computer Systems to Digital Communication Networks,

5. TYPE OF REPORT & PERIOD COVERED

(9) Final Report. May 75 — Jun 76,

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

(10) Mario Barbacci, [redacted] Lih/Chang, Samuel/Fuller, Ashok/Ingle Navindra/Jain

8. CONTRACT OR GRANT NUMBER(s)

(15) DCA J00-75-C-0066

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Department of Computer Science,
Carnegie-Mellon University
Pittsburgh, PA 15213

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

PE 33743K

11. CONTROLLING OFFICE NAME AND ADDRESS

Advanced Systems Concepts Branch, R740
Defense Communications Engineering Center
Reston, Virginia 22090

(11)

12. REPORT DATE

26 Jun 76

13. NUMBER OF PAGES

~~130~~ (12) 153p.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

Advanced Systems Concepts Branch, R740
Defense Communications Engineering Center
Reston, Virginia 22090

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Distribution unlimited

18. SUPPLEMENTARY NOTES

None (4) This report describe work done to

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multiprocessor
Integrated Switch
Time Division Multiplexing
Slotted Envelope Network
Simulation
Circuit Switching
Packet Switching
Voice Communications
Data Transmission

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Defined and implemented an experimental facility that emulates the behavior of a single node in an integrated circuit-packet-switching communications network: (1) Identified the functional characteristics of the system, (2) Developed and implemented a task decomposition that performs the different functions of an integrated switch, (3) Developed theoretical switch performance models and (4) Developed a reliability study of the hardware components.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

403 081

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

3.- Effect of Periodic Maintenance on Reliability

- 3.1 Life of an Unmaintained System
- 3.2 Life of a Maintained System
- 3.3 Redundant Systems with Maintenance
- 3.4 Life of an Unmaintained, Redundant System
- 3.5 Life of a Maintained, Redundant System

Abstract

The work performed during the first year of the contract has resulted in the definition and implementation of an experimental facility that emulates the behavior of a single node in an integrated circuit-packet-switching communications network. There are several major components in this study: We have identified the functional characteristics of the system, we have developed and implemented a task decomposition that performs the different functions of an integrated switch, we have developed theoretical models of performance for the switch, and finally, we have developed a reliability study of the the hardware components of the system. The report does not attempt to present results as a closed set of conclusions. The main product of the first year has been the definition of the problem task, the development of the experimental facility, and performing related theoretical studies; much work remains to be done in both the experimental and the theoretical aspects of the project. These will be the object of the second year contract. The continuation effort will not only elaborate on the first year results but will also expand into other areas. These areas will be identified throughout the report.

Chapter 1 describes the characteristics of the integrated switch and the experimental facility implemented on C.mmp. This chapter presents the results of running several experiments with various mixes of Class I and Class II traffic. Chapter II describes the analytical models for the integrated switch. Chapter III describes the reliability models.

Carnegie-Mellon University Report

Chapter 1

A Multiprocessor Implementation of an Integrated Switch

TABLE OF CONTENTS

CHAPTER 1 PAGE

1.1. User Requirements

1.1.1. Circuit and Packet Switching

There are basically two standard techniques currently used in communication networks: line-switching and message-switching. In line-switching the caller first requests a connection to another subscriber. After the channel is established (usually after the receiver has acknowledged the request and is ready to communicate) information can then be transmitted over the allocated channel. In message-switching, the sender starts transmitting information as soon as a path to an intermediate switching node is obtained, thus allowing the use of adaptive routing techniques. Messages are temporarily stored in the switching nodes until completed. Whenever a path is available, the message is relayed to a neighbor node along a path to the final receiver. This technique is also known, for obvious reasons, as Store-and-Forward Message-Switching.

The choice of which scheme to use depends on the characterstics of the communication traffic. Line-switching (also known as circuit-switching), is more advantageous in those situations in which the transmission time for a message is large compared with the overhead needed to establish and break down the connection. It has the advantage that no buffer space is needed along the intermediate nodes. When the size of the messages decreases, the overhead is relatively larger and message switching is preferable.

To simplify error recovery and to smooth the volume of traffic over the network, in certain message switching systems, messages are split into fixed size

"packets" which are transmitted throught the network. In packet-switched systems, individual packets extracted from a message might travel different routes and thus experiment different delays, necessitating a reassembly phase to be performed at the receiver end.

Packet switched networks present a potential cost advantage for some mixes of traffic over the use of dedicated lines due to the time sharing of the lines among many simultaneous users. Having this type of guaranteed connections results, however, in variable throughput and delay as the load on the network varies. This is in contrast with the behavior of a circuit switched network: when the network is loaded, the connection time might increase (sometimes to the point that requests for connection might be rejected) but once a connection is established the throughput and delay remains constant.

In a typical circuit switching network such as the telephone system, the fact that (usually) only one of the speakers is active at a time, combined with the frequent silence gaps tends to make inefficient use of the channel capacity. For the entire network the situation is even worse: most of the lines are idle most of the time in order to provide the excess capacity that must be available to achieve a specified grade of service. Packet switching on the other hand, allows a greater flexibility in the utilization of the network. The channels are used only for the time of the actual transmission, the end-to-end link being provided by a logical channel, not a physical one.

1.1.2. Integrated Switches

There are three major arguments which indicate the desirability and usefulness

of an Integrated Communications System. These are: Cost reductions, survivability, and homogeneity of service.

Cost Reductions

Use of separate networks for data and voice imply separate transmission facilities, which often travel parallel geographical paths. The increased cost comes not only from the transmission equipment direct costs combined with the loss of the economy of scale, but also from the increased number of channels required to satisfy the degree of service requirement for voice and the maximum delay allowed in interactive transactions. Communication networks are designed with certain amount of extra capacity in order to accommodate peaks of traffic. The percentage of extra capacity actually increases with smaller systems. [Cov75] shows that for a given quality of service (0.02 probability of rejection of a call), 15 channels are required to transmit 9 Erlangs of traffic using separate networks (5 Erlangs of Voice traffic plus the equivalent of 4 Erlangs of data traffic), while using an integrated network only 10+ channels are required without significant degradation of either type of service. The key factor in this saving is the dynamic allocation of the network total bandwidth to the traffic demand.

Survivability

A second major advantage of an integrated network is the capacity of the network to respond to crisis situations. Using separate systems, a cripling accident will reduce the capacity of the affected network without being capable of utilizing any spare capacity available in the other network. An integrated system can respond instantaneously to a crisis by such actions as allocating the surviving capacity to the highest priority traffic, preempting low priority calls, suspending services for low

priority data packets, rerouting circuit switched communications, etc. Emergency adaptability, as opposed to predictable allocation of traffic as a result of average busy hour demands, is a vital requirement in defense communications.

Homogeneous Service

One of the objectives of the integrated approach is to provide services to the largest possible community of users. Using separate circuit and packet switching networks would not allow interaction between subscribers connected to separate systems. Although it could be argued that current usage patterns do not require interaction between say, a telephone subscriber and an interactive computer, future applications can be envisioned in which such interaction is desirable [Cov75]. For instance, we can visualize a query system in which information is stored in a computer data bank and is accessible to remote computers as well as users connected to the network through a telephone. Current research on Speech Understanding Systems [Red76] indicates the feasibility of real time man-machine voice interactions, with task restricted vocabulary of a few hundred words but with a high degree of accuracy (better than 99% for single word recognition, slightly less accuracy for sentences).

Although voice transmission is being used as the canonical example of circuit switched traffic, there are other types of real time information transfer, e.g.: remote control instrument monitoring, graphic and television images, facsimile, etc. all of which share the same requirements for constant throughput and minimal delay.

1.2. Integrated Switch Modeling

1.2.1. The SENET Scheme

The integration of circuit and packet switching is based on the transmission of information using a Time Division Multiplex (TDM) scheme first proposed in [Cov75]. In this scheme, time slices of fixed size - a frame period - are transmitted synchronously over high speed lines between two adjacent switches in the network. The frame acts as an envelope for smaller time slices of variable size, each acting as a slot for transmitting a "packet" of information.

In this Slotted Envelope NETwork (SENET) scheme, frames are transmitted synchronously between adjacent nodes and a circuit switching subnetwork can be defined by the reservation of fixed slots inside a frame. Since the slots carry the same amount of bits every frame period, beginning at exactly the same point in time, this is equivalent to the reservation of dedicated channels between the two adjacent nodes. The rest of a frame can be allocated on a demand basis. If a suitable protocol is defined for the proper interpretation of the information transmitted on this part of a frame, a packet switching network can be easily implemented.

Traffic Types

The primary driving force in the design of a communications system is the character of the information it must handle. Based on existing communication networks, three different classes of traffic can be identified:

Class I.- Characterized by long transactions requiring continuous real time response (voice, video, facsimile). This type of traffic can be transmitted in the synchronous portion of the frame. They are representative of transactions transmitted in a circuit switching "network".

Class II.- Characterized by short discrete transactions requiring near real time response (interactive data). This type of traffic can be transmitted by dynamic allocation of slots in the asynchronous portion of a frame. They are transmitted in a packet switching "network".

Class III.- Characterized by long transactions requiring neither continuous nor immediate response (bulk data). This type of traffic can be transmitted, as the previous class, on the asynchronous portion of a frame.

The details of a frame are shown in Figure 1.1. Starting at 12 o'clock a certain number of bits are reserved for CCIS (Common Channel Interswitch Signaling), followed by a Class I region containing the real time traffic. The end of the Class I region is indicated in Figure 1.1 at 5 o'clock. Class II and III regions containing the interactive and bulk traffic occupy the rest of the frame. (Unless we make it explicit, we shall make no distictions between Class II and Class III traffic. We will use the term Class II to indicate both interactive and bulk data).

The differences between the requirements and characteristics of the above classification of traffic imply different types of service:

Class I traffic is either accepted or rejected, with short connection delays and without error control. Class I traffic requires low delay and a constant throughput in order to maintain the intelligibility of the message. Class II traffic is always accepted but may incur a system delay, with short connection and cross-network delays. The traffic is characterized by bursts of information followed by "waiting" periods, requiring a high degree of reliability. Class III traffic is always accepted (although the network might temporarily suspend this type of service), with longer connection and cross-network delays than the previous class. In Class III traffic, variations in the delay of individual packets is not important and this can be used to reduce the impact of the long periods of high traffic volume. As in Class II traffic, a high degree of reliability is required.

The above discussion summarises the top level characteristics of an Integrated Switching Network. For the purposes of the project, however, we had to take a closer look at lower level considerations that will play a role in the actual implementation of a SENET system. The remainder of this section outlines the major characteristics of a possible implementation.

Class I Reservation Tables

The real time requirements of Class I traffic dictates that it be processed in a special fashion by the integrated switch. Since voice carries a significant amount of redundancy, a larger error rate can be tolerated in most applications (secure voice transmission presents lower tolerance to errors). By eliminating or reducing the need for error control, the transmission of Class I traffic can be performed in a straightforward manner. The establishment of a logical circuit between two subscribers is reflected, on each switch along the path, in the updating of Class I Routing Tables internal to the integrated switches. These tables indicate, for each slot in the Class I region of an input frame, both the output frame (output channel) and slot position reserved for the logical circuit, as shown in Figure 1.2.

The reservation of entries in the routing tables is performed during the establishment of the communication and remains in effect until the communication is terminated. The routing tables also contain information about the slot type (voice, video, etc.) and precedence (Flash, Flash Override, etc). These two parameters are used to allocate, reserve, and preempt the logical channels. The reservation of the slots along the communication path guarantees a fixed bandwidth for this type of traffic. As traffic requirements vary during the day the portion of a frame reserved for Class I traffic can be reduced or expanded accordingly.

Non-Homogeneous Links

The input and output links in the network need not necessarily be homogeneous. For a given time window length, say 10 milliseconds, the length of a frame varies according to the capacity of the link. Thus, for a T1 carrier, a frame contains 15,440 bits. For slower carriers, the frame will be accordingly smaller.

Allowing the use of lines with different capacities permits the use of similar nodes in different capacities in the network. They can appear as switching nodes, connecting high speed trunk lines, as part of the access system connecting and concentrating large numbers of low volume users, connecting HOST computers into a communications network, etc.

Asynchronous Behavior

The scheme does not assume the existence of a master network clock. Each link transfers frames at a constant rate (1 frame/time slice) but the links are not necessarily synchronized among themselves.

Requiring a centralized network clock has a serious impact on the reliability of the network. In an asynchronous network each link works independently from all other links. There is no central critical component (a clock) whose failure will bring the network down. Each link between two nodes constitutes an isolated entity whose rate of failure has no impact (other than perhaps on the volume of traffic permissible) on the network. This approach also allows the presence of transient errors in a link which might possibly throw it out of step. Link protocols and error detection and reporting are, the responsibility of the two nodes interconnected by the line.

Line Interface Devices

The line functions (detection of frame and packet headers, detection of special bit patterns or flags, Cyclic Redundancy Checks, etc) are performed by special purpose

hardware devices. The actual (physical) input and output operations are performed by these Line Interface Devices (LID) using their Direct Memory Access (DMA) capabilities.

The presence of hardware devices to handle line error detection frees part of the node processing capability. From the LID, the frame is loaded directly into memory and the event is indicated to the integrated switch processor(s). Having special line handling devices allows the detection of errors on a packet per packet basis. If error detection were performed over the entire frame, a single error anywhere in the frame might make it invalid and therefore the frame might have to be retransmitted, an unacceptable situation from the point of view of Class I transmission.

Bytes as Transmission Units

We assume an 8-bit byte as the smallest unit of information transmission. Thus, the integrated switch processors will treat a frame as a vector of 8-bit characters. Headers, packets, and frames are therefore assumed to contain an integral number of characters.

Although the actual transmission of information is done on a bit serial basis, existing processors rarely use the bit as the unit of directly addressable information. A bit addressing capability would imply a larger address field in the instructions, a serious penalty to pay for a rarely needed capability. This is not an unreasonable assumption. Packet sizes are typically several hundred bits long and the possible waste (if any) will be a small fraction of the traffic.

Packet Transmission Format

The use of the Advanced Data Communication Control Procedures (ADCCP) [ADC75], or variant thereof, seems reasonably well-suited to this application. Each packet to be transmitted is augmented, as in ADCCP, by hardware-generated header

and trailer fields. The header field identifies the start of a packet (by a unique flag-character) and passes control information to the Line Interface Device (LID) at the destination. The trailer field contains the cyclic redundancy field followed by the flag-character to indicate the end of the packet. By so delimiting the packet with unique flag-characters, error detection becomes straightforward.

Figure 1.3 illustrates a possible transmission format along the lines of the above discussion. The figure shows the flag-characters being used as idle characters indicating an idle transmission line. Other conventions are possible. By use of a bit-stuffing technique the flag-character can be guaranteed not to occur within the augmented packet. The reverse operation at the destination recreates the original packet.

The design thus described is self-synchronizing in the sense that there is no need for explicit synchronizing signals or procedures. The source node can send a packet anywhere inside the asynchronous portion of a frame. Similarly, the destination detects the start of a transmission simply by noticing the absence of a flag-character. Because of this independence between source and destination ends of the lines, only local clocks at the source ends are necessary to time the frames.

Class I Transmission

Since Class I slots are small (say, 80 bits for an 8 Kbps vocoder channel [see For75]), sending each as a separate packet would entail a great deal of overhead. Indeed, the main reason for using separate packets for Class II traffic is error detection, which is not a concern with Class I transmissions. The reasonable conclusion is to send the entire Class I portion of each frame as a single packet. This has the advantage of minimizing the number of overhead bits required (a single header/trailer

pair). However, it is important that this Class I packet should always be accepted by the destination LID, even if a transmission error is detected. What is needed is a type field in each packet header which identifies the packet as either Class I or Class II. (In a modified ADCCP, the "control" field of Figure 1.3 could be used for this.) The solution to the problem is then to have the destination LID intentionally ignore any possible error indication for Class I packets. As a consequence, control information used to establish or break Class I communications must be send as Class II packets. Thus, a frame can not be dedicated exclusively to Class I traffic, some small partion must be reserved for Class II control packets for this purpose.

Frame Generation and Timing

According to the original Coviello and Vena concept, new real-time data appears say, every 10 milliseconds (in a T1 carrier this corresponds to 15,440 bits/frame). Hence, every 10 milliseconds a Class I packet must be transmitted. The start of a frame would simply be indicated by receipt of each Class I packet header. No explicit start-of-frame marker would be required. Whenever a new Class I packet begins to arrive, the LID can interrupt its attached switch processor to announce receipt of the previous frame. In Figure 1.4 for example, a schematic "snapshot" of packets traversing a transmission from right to left, the destination LID would be storing frame 1 into a memory buffer (via DMA) as it is arriving. When packet I_2 begins to arrive, the LID would signal an interrupt to indicate receipt of frame 1. When I_3 begins to arrive, receipt of frame 2 would be signalled, and so on. In other words, a frame consists of a single Class I packet followed by zero or more Class II packets. The intervals between Class I packets are filled, either partially or completely, by Class II packets.

1.2.2. Task Decomposition

Before going into the details of the functional decomposition of the system we must provide an overview of C.mmp and its architecture since it plays a central role in many of our design decisions.

C.mmp (Figure 1.5) is a multiprocessor system designed and built at CMU under the sponsorship of the Advanced Research Projects Agency (ARPA) of the Department of Defense. The system consists of (up to) 16 miniprocessors (DEC PDP-11/20 and PDP-11/40) connected through a central crosspoint switch to a large shared memory (up to 32 million bytes). In addition, each processor has a small amount of local storage for private code and data. The use of a large shared memory makes it possible to define multiprocessor sub-systems by the proper allocation of the storage to small clusters of cooperating processors. Both inter- and intra-cluster communication and synchronization can be implemented by sharing buffers and interlocks acessable by the individual processors. Although C.mmp also provides a limited interprocessors interrupt facility, it was not used in our system for reasons explained in Section 3.3.

The emulation of a single integrated switch is performed by a cluster of processors sharing both code and storage. Other processors, external to the cluster are used to simulate the outside world (the network) and are in charge of feeding multiple frame streams (simulating multiple, asynchronous, input channels) and collecting statistics about the behavior of the integrated switch.

Inner and Outer Loop Functions

The functions to be performed in an Integrated Communications Network can best be studied if we recognize a classification in terms of their impact on the global network behavior and in terms of their processing requirements on the individual

switches. The following decomposition separates those functions that are performed by the integrated switches in order to establish and maintain communication with the other switches of the network (the relay functions) from those that are performed in order to establish and maintain the accessability to the network by the local subscribers (the regional functions). The classification can be further refined by identifying those functions that are continuosly performed as part of the communications task from those that are performed infrequently, as a result of exceptional conditions (e.g., error detection and recovery) or during the initialization/termination phases of a communication between two subscribers.

Functions marked with an asterisk (*) are "outer-loop" functions, i.e., functions which require negligible execution time compared to the "inner-loop" functions (those not marked with an asterisk). Only "inner-loop" functions are included in the first version of the system.

Relay Functions (Trunk lines)

(1) Routing Selection.

- Choice of output line to send message out on, including messages originating at this node.

- Essentially table-driven at this level.

(2) Message Acknowledgement.

- Acknowledging received messages.

- Timeout on expected acknowledgements for sent messages.

(3) Channel Discipline.

- Frame decomposition and assembly.

- Allocating/compacting real-time message slots in frames.

(4*) Maintainance of Routing Tables.

-Timing of message delays to neighboring nodes for adaptive routing.

(5*) Hardware/Software Fault Detection.

- Intra-node detection: Timeout, NXM, data structure integrity check, reliability analysis.

- Inter-node detection/correction: Status of other nodes as viewed over interconnection links; external load/restart capabilities.

Regional Functions (Access lines)

(1) Analysis/Validation/Generation of Packet Header Fields.

- Precedence check: is this terminal authorized to use this level of precedence?

- Security authorization check: is this terminal authorized to send/receive messages of this security level?

- Comparison of header security keys: do they match?

- Segment count: check for any missing or out-of-sequence and appropriate actions if so.

- Generation of any necessary header information for originating messages.

(2) Flow Control Functions.

- Access denial: refusing originating traffic for categories not being accepted by destination (e.g., because of congestion)

- Queue-length monitoring, to detect congestion.

- Purging of low category/precedence traffic from queues if necessary.

- Delay of acknowledgements if congested.

(3) Logical Channel Control.

- Opening/closing of logical channels, especially for real-time connections.

(4) Precedence Handling Functions.

- For high precedence messages.

(5*) Identification/Validation of Terminals.

- Table lookup, possible password check, etc.

(6*) Compatibility Conversions.

- Character translation, mode/format conversions.

(7*) Maintainance of Regional Tables.

- Line tables, terminal ID tables, logical channel tables, etc.

(8*) Hardware/Software Fault Detection.

(9*) Statistics Gathering/Reporting.

The scope of the first year project centers around the inner loop functions and their emulation on C.mmp. Thus we only deal with an isolated node although the presence of the "outside world" (i.e. network) is a driving force. This will be elaborated later, when we discuss the Script Generator and the Statistics Collection.

Software Organization

The handling of real time and high priority traffic indicated that a conventional multiprogramming approach could not perform the mission. The time required to switch contexts (in the order of several milliseconds in Hydra) could very well prevent a processor from fulfilling a request within an acceptable time. However, the use of homogeneous processors and shared storage permited us to implement the system as a set of procedures directly executable by any processor, operating on shared data. Thus, as in multiprocessing, no processor is dedicated to any particular task. Context switching is however, replaced by a faster task switching scheme described in the next paragraph.

The scheduling of the tasks is performed via a set of priority ordered queues (Figure 1.6). Some queues are dedicated to a specific task (for instance, routing). Others are dedicated to families of related tasks. In the idle state, processors are continuously executing the scheduling task, namely, scanning the queues according to their priorities. When a non-empty queue is found, the scheduling task takes the first

element of the queue and the processor executes ("calls") a procedure that implements the task that was found in the scheduling queue. The execution of a task's procedure might result in the creation and queueing of other task requests. When the task's procedure is completed, the processor resumes the execution of the main loop, the scheduling task.

Class I Traffic Processing

As indicated before, voice and other real time traffic carries a significant amount of redundancy and a larger error rate than on data can be tolerated. This suggests that performing error-checking and generating acknowledgements for Class I packets can be avoided. However, the establishment and maintainance of a Class I channel is a critical requirement thus, the control messages interchanged between network nodes in order to establish, re-route, and break-down Class I connections must be thoroughly reliable. These type of messages are best transmitted as Class II packets. It should be noticed that given the long holding time for a typical voice channel (180 seconds or 18,000 frames in a 10ms/frame network), the few messages that might be needed for control purposes represent a negligible amount of extra traffic.

When a frame arrival is detected by the input LID, an event notice is posted in a pre-assigned location. As soon as a processor becomes available, after completion of its current task, the arrival notice is removed and the processor initiates the task of decomposing the frame. This task gives special consideration to Class I traffic, i.e. to minimize the switch delay, transmission of Class I traffic is given precedence over most Class II related tasks.

The handling of incoming Class I traffic by the switch program at a trunk node is basically a "scatter-write" operation, where the Class I slots arrive in a contiguous

buffer but are then copied piecemeal to various output buffer locations, as determined by the Class I Routing Tables.

Independent of the "scatter-write" implementation, the primary design goal is to minimize the delay time through a node. However, there are some pitfalls that one must be aware of in constructing such a design. These involve the possibility of slippage or overwrite of Class I slots. The former problem consists of the introduction of random "silence" gaps when a Class I slot misses its output frame and is delayed one extra frame period in the switch. The latter problem occurs when a Class I slot is overwritten by the slot arriving in the next frame and before the output frame has been transmitted. These two problems are studied in more detail in Appendix I.

Class II/III Traffic Processing

During frame decomposition, the individual Class II packets are isolated, and pointers to each individual packet are posted in a Class II copy task queue. When this frame decomposition is terminated, the processor returns to the available pool and competes with other idle processors for pending tasks.

Figure 1.7 depicts the sequence of tasks in the integrated switch system. Q_0 is the Input Frame Queue, Q_1 is the Class I Copy Queue, and Q_2 is the Class II Copy Queue. The Class II copy tasks consist of copying the data packets into individual buffers (due to higher reliability requirements, data packets must be kept in the switch after they are retransmitted while waiting for an acknowledgement). After a packet has been stored in a buffer, different tasks might be created depending on the nature of its type, precedence, and final destination (Q_3 and Q_4). Some Class II packets may convey line control information or acknowledgements. These packets will be either forwarded to other switches or a specific control function will be performed and the

packet will then be deleted. For data packets with a foreign final destination, routing and acknowledgement generation tasks are created. Packets addressed to a local HOST might instead be placed into a re-assembly queue. The routing tasks are implemented on Q_5 and Q_6.

Queueing Models (An Introduction)

Although the theoretical performance model of the SENET implementation on C.mmp is fully elaborated in Chapter 2, it is appropriate to make a few remarks about the queueing characteristics of the integrated switch as they relate to the preceeding discussion.

Class I traffic is a fully deterministic process. The combination of a relatively long holding time (several thousand frames) together with a constant service request (essentially looking up the Class I Routing Tables and copying the Class I slot into the appropriate output frame) allows us to simplify the modeling of this part of the system.

Class II packets present a totally different picture. The need for error checking, generation of acknowledgements, precedence driven processing, computation of routing information, management of buffer space, etc., result in a truly random service request for each packet that arrives at the integrated switch.

The combination of Class I and II traffic results in a system in which the inter-arrival time of the request for service on the different queues in the system is neither exponentially distributed (a common assumption in theoretical studies) nor deterministic. For each frame there is a constant level of Class I traffic requests and some random level of Class II traffic requests. Although there are no simple models to analyse such a system, several approximations were developed at CMU and are presented in Chapter 2.

1.3. Experimentation Utilizing the Physical Model

1.3.1. Experimental Design

The general design of the experiment on C.mmp uses two processors outside of the switching node for both the generation of the script and the analysis of the results. The script generator process (described below) generates the incoming frames for each of the channels and stores them in a set of buffers reserved for this purpose. Currently, there is a ring of eight buffers on each channel to allow for variation in the speed of the script generator process (a frame with few packets takes less time to generate than one one with many packets).

The analysis process is also outside of the simulation per se. It's task is to examine all the Class I slots and all the packets send on each frame, for each output channel. Delays of packets and Class I slot transfers are computed for later calculation of mean and standard deviation of various categories of packets.

In addition to the Script Generator process and Analysis process, there is one other component of the simulation: a KWP-11 programmable timer which is supported under HYDRA to provide precise timings of frame durations, arrival times, etc. Because of the limitations of running under the HYDRA operating system, as explained in Section 3.3, the simulation program is not allowed to field the timing interrupts directly. Instead, a form of polling is used, where each of the "slave" processors (the ones carrying out the simulation) frequently check to see if an interrupt has been signalled. Given a reasonable number of slave processors, the delay before an interrupt signal is recognized can be made reasonably small.

The overall configuration can be symbolically described as "1+1+N" where "N" represents the N slave precessors in the simulation, and "1+1" represents the Script Generator and Analysis processors.

The original thought in the simulation was to pre-generate a large script and simply read it in as it was required. However, since secondary memory would be required for this, there was no satisfactory way to guarantee that (a) segments of the script could be read into main memory in time, and (b) the operating system would not usurp some of the processors for large periods of time (tens of milliseconds) to perform the input/output operations. Hence, it was necessary to ensure that the simulation would be entirely compute-bound and not do any input/output during an actual run. This required that the script and analysis be performed " on-the-fly". The main drawback with this approach is the limitations of how fast the script can be generated, and how fast the results can be analyzed and accumulated. More will be said on this later.

The Script Driver

A separate processor is dedicated to simulating the traffic to the node. This processor continously runs the Script driver program generating the traffic according to specifications told in a dialogue during the initial set up. The system timings are kept with a programmable hardware clock in conjunction with a fast interrupt routine which sets software interrupts, indicating arrival and departure of the frames or other events. These interrupts are attended to by system processors hunting for work. In order to meet the randomly varying amount of required work, the driver does not work on a frame timing basis. Instead, for each line, there is a ring of eight buffers and a variable pointing to the current frame , and the script driver (after taking a

headstart) continuously fills empty buffers with new frames. Each frame carries a sequence number which apart from identifying frames is also used for finding its arrival/departure times, since these occur at a fixed specified interval. The actual message is of no concern to the node, and no attempt is made to generate 'garbage' messages in order to conserve space and time. Only headers are processed and generated. The format of frame and headers is described in Figure 1.8.

Node/Network Parameters.- The parameters that can be specified for node and the traffic nature are as follows:

Number of lines connected to the node. A "soft" limit of 4 channels is assumed, mainly for buffer allocation purposes. If during the experiment runs need is felt for having a bigger number, it can be easily accomplished.

Frame duration (typically 10 msec.), relative arrival and departure times of frames on every line. This information is used for initialising and directing the hardware clock routine as to when and which interrupts to post.

Line Parameters.- Each line is characterised by:

Speed in Kilobauds.

Composition of the real time packets (i.e. length and destination of all real time slots). In this set of experiments the duration of the run is kept much smaller then typical holding times of real time traffic. Hence the reservations for these are preset and are not changed during the run.

Composition of the Class II and III traffic region. There are twenty one levels of priorities in Autodin-II specs, however such a large number of levels are of no interest for experiments, since they make design and analysis of experiments unnecessarily difficult. Therefore only six classes are implemented, and during analysis the larger set can be meaningfully mapped onto them. The six classes are Control, Realtime, Data High, Data Low, Bulk High, Bulk Low. The fractions indicated are used as discrete linear distribution for determining type for a packet.

Direction of traffic flow. That is destination of packets incoming on this line. Fractions are indicated for every other line, which are used as discrete linear distribution for deciding the destination of packet at random.

Average number of data packets per frame and their distribution. A poisson distribution is a good choice for this.

Length of packets and their distribution

Additional Parameters.- Apart from these above line characteristics the following also need to be specified:

How long the system should run .

How long the system should run before steady state is obtained and statistics collection can be started.

Times for statistics collection relative to frames arrival times.

Seeds for the random number generator.

The script driver prompts for the above information in the beginning. During this phase the script driver prompts the user for the emulation parameters and allows the user to set, display, and modifify parameters specified for a previous experiment. An example of a work session in Appendix II illustrates its exact use and is self explanatory. A consistency check is made as far as possible. In case of meaningless specs, error messages are issued and for doubtful cases warnings are given. Thus specifying minimum length of a packet greater than its maximum length would cause an error message and cause the question to be repeated. Specifying a large average number of packets on a slow line would solicit a warning. A run time record of the nature of traffic actually generated is kept and later the statistics in terms of the traffic intensity, average number of packets/frame, average length of packets, fraction of traffic in various classes, etc. is reported.

Operation of the Experiment

The experimental results are obtained from series of discrete "runs". Each run represents a completely independent simulation; the input parameters can be similar to

those of other runs, or widely different. The experimentor can vary the parameters to create a particular situation worthy of study. The process is interactive: the experimentor can observe the results of one run and immediately apply that information to the next run.

The cycle of operation in running the simulation is as follows. Initially, the experimentor must interact with HYDRA to create the necessary processes, I/O connections, etc. This is not repeated during the remainder of the session. Once the initial set-up and allocation has been done, the program asks for input. Appendix II (on the Script Generation parameters) shows an example of a dialogue with the input-section of the program. It is not necessary to always type in a whole net set of data each time -- if the parameters are similar to the previous run, the input-routine allows one to only specify items that need to be changed.

After input, the experimentor allows the program to begin, and waits until it returns to the terminal to ask where the output should be sent (eg, line printer, terminal, etc.). After output is completed, the program is ready to accept new data (or changes to the last data) and run again.

Explanation of Output

Output from the program is given in the form of tables, which require some explanation. The first tables to be printed concern the Script Generator process. It is useful to know exactly the attributes of the script that was generated. For one thing, this serves as a check that the Script Generator is performing correctly and generating data that corresponds to the parameters read in. In addition, measurement of the generated script is necessary to obtain some of the parameters which cannot be specified directly as input. (For example, average data capacity used cannot be

specified directly as a distribution parameter, since it also depends on how often a packet must be delayed until a subsequent frame because of lack of room -- this cannot be calculated ahead of time easily.)

Figure 1.9 shows the first page of program output. The first section simply displays the input parameters. This is followed by the actual measurements on the script: average number of packets sent on each input channel, and fraction send to each of the output channels; average packet length (in bytes), fraction of data capacity used on each input channel, and total frame capacity used (including Class I traffic). Finally, the breakdown of the data packets sent by priority is displayed.

Figure 1.10 illustrates the output from the Analysis process for a sample run. The first table, entitled "Realtime Delays" (ie, Class I traffic) is basically a histogram showing the total number of Class I slots transferred within n frames. It is clearly desirable to have the cross-node delay for Class I traffic be as small as possible, but there are dangers associated with trying to make the delay too short (see Appendix I on Class I slot slippage/overwrite). In the simulation, three buffers are used for the Class I traffic. This is necessary to avoid the possible dangers mentioned, but it means that the cross-node delay will always be a duration of two frames. In addition, any relative skew between incoming frame and outgoing frame is added onto the two frames' delay. Hence, all the slots from a specified channel should be transferred in exactly two frame's delay (plus skew, which can range from 0 up to almost a frame in duration, depending on the relative frame arrival time compared to the frame departure time). Class I slots transferred during any other frame indicate an error or exceptional condition, such as the node program being unable to meet the deadline requirement. Similarly, the "Empty Slots" column indicates the number of empty Class I

slots seen by the Analysis processor. This is another indication that the simulation processors were unable to meet their deadline requirements for Class I processing.

The second table is a bit more self-explanatory. Each Class II/III packet receives a time-stamp upon entering the system from the Script Generator. The Analysis process, when it sees each packet at the output end, computes the cross-node delay for that packet, and keeps a running sum of the delays and squares of the delays (for calculation of standard deviation). The results are tabulated by priority. In addition to the standard statistics, an attempt is made to calculate the buffer-lifetimes used by the data packets (ie, packet length*delay). This lifetime does not currently include any estimation of the time required for a return ackowledgement (which is a major factor).

A number of checks have been included in the program to test whether the real-time deadlines are being met. Whenever critical tasks are not performed in time, certain counters are incremented. These are displayed just before the delay tables, and have the following meanings. "Noverrun" gives the number of times that the Script Generator was unable to keep up with demand for incoming frames. The major factor here is the number of packets to be created in the frame, and thus how many times the random number generator must be invoked. "Vstattooslow" represents the number of times that the Analysis routine had to give up on analysis of the voice (ie, Class I) data because the buffer had just become the current buffer being loaded. "DPoverrun" is the number of times that the available space for outgoing packets was completely filled. Currently, there are a total of six "CCVs" (Channel Command Vectors) which serve as a ring of buffers, much as the Class I buffers are used. Each outgoing frame consists of the appropiate Class I buffer followed by the list of data

packets on the appropiate CCV. When a packet is to be sent and all the CCVs are full, then DPoverrun is incremented. Finally, "VCoverrun" indicates the number of times that the voice, or Class I, copy task was unable to meet its deadline before the specified buffer began to be sent out.

Figure 1.11 shows the remainder of the output from a run. The first section shows the breakdown of tasks performed by each of the "slave" processes. The tasks are shown across the top, and the number of times each process looked at that task queue, and the number of times that that process found a task to do, are tabulated. The ratio of the two is also calculated and printed.

The second section gives average queue lengths in the system, sampled on a regular basis (once per frame-duration). An idea of the amount of interference between the processes is obtained by examining how often a queue was locked by a process when access was attempted by another process. Basically, the tasks table and the queue-length table are not directly related to the performance of the node simulation, but they can offer information on the internal workings of the simulation.

1.3.2. Experimental Results

Data Traffic

System Parameters:

Line Speed: 1.544 megabits/sec. (T1 carrier)
Frame Length: 10 milliseconds
Number of Channels: 2 (full duplex)
Number of Processors: 3 (PDP-11/20)

Frame Departure Time: 0 msec.
Channel 0 Frame Arrival: 1 msec.
Channel 1 Frame Arrival: 9 msec.

In the first series of experiments we studied the limitations of a small switching node (3 miniprocessors). The experiments were designed to compute the average intranode packet delay in the absence of real time traffic. I.e. we were measuring the ability of the configuration to respond to various degrees of traffic intensity and the speed with which Class II and III packets could be transmitted between two incoming and two outgoing lines. The traffic was assumed to consists of two types of packets, each acting as representative of Classes II and III respectively. The relative volumes of each traffic type were set at 20% (Class II) and 80% (Class III). Moreover, the packet lengths were drawn from the same uniform distribution, bounded beween 200 and 2200 bits respectively

It was observed in preliminary runs that one of the limitations of the system would be the ability of the C.mmp processors to move the data packets from an incoming frame buffer into the individual packet buffers. If we consider that a PDP-11/20 takes in the order of 5 microseconds to perform a memory-memory word movement, just copying the information contained in a full frame would take: 15440bits/frame*1word/16bits*5microseconds/word = 4825microseconds. In other words, almost half the time available (10 milliseconds) could be spent just copying the information, not counting the overhead associated with the queue and space management. If we add to this figure the time needed to process the packet (precedence identification, routing, etc) and the global overhead (polling interrupts, computing queue statistics, etc) it becomes apparent that the average number of packets/frame together with the copying mechanism used are the major independent parameters. The results of the experiment are shown in figures 1.12 through 1.14. In these figure the Y-axis indicates the average packet delay for each class (two curves

are shown in each figure) and the X-axis indicates the average number of packets/frame.

Three different "copying" schemes were tried. Figure 1.12 shows the result of using the standard PDP-11 MOV instruction in which one word is copied between two memory locations. The figure shows that although high priority traffic receives a fairly constant degree of service over a variable traffic volume (represented by the average number of packets/frame), low priority traffic tends to go unattended for longer and longer periods of time, represented by the exponential increase in the average delay. The figure shows that beyond 4 packets/frame the system performance deteriorates drastically.

Figure 1.13 shows the result of using a better (hypothetical) block transfer instruction that might be implemented on the PDP-11 processors. We were conservative in our approach and assumed a speed-up factor of 4 (roughly equivalent to having a 1.25 microsecond MOV instruction, all other instructions being the same). The figure shows a slight improvement over the previous result. The system is capable of handling an average of 5 packets/frame before the low priority traffic delay becomes the limiting factor. It can be observed that the delay for high priority traffic remains constant over the range of traffic loads.

Figure 1.14 shows the upper bound in the performance of the PDP-11/20's used in the experiment. Here we made the simplifying assumption that specialized hardware, running concurrently with the integrated switch processors, could take care of the mundane task of moving blocks of data between memory locations. In other words, we supressed the actual copy operation (although the overhead of setting up the copy loop was still present and accounted for). The figure shows a slight

improvement over the previous results. The system could now handle an average of 6 packets/frame before the average delay of the low priority traffic became impossible to handle.

In all the experiments the final limit was reached when the arrival rate so outgrew the service rate that critical resources like queue elements and packet buffers became exhausted.

Skew Impact

One of our early asumptions about the implementation of a SENET was that no synchronization between input and output lines was needed or advisable. In this experiment we attempted to measure the impact of the skew between the frame arrival and departure times. Figure 1.15 shows the effect of shifting the input frame arrival time given a fixed frame departure time. The figure shows the variation in the average packet delay for a load of 4 packets/frame (average). The packets themselves were drawn from the same distributions as in the previous examples.

Figure 1.15 indicates that varying the time gap between the frame arrival and departure times has more impact on the high priority traffic. The shape of the high priority delay curve can be explained as follows: For small values of the skew (1 or 2 milliseconds) very few packets can be processed in time to be transmitted on the very next output frame. Thus, the delay grows linearly with the skew. For skew values between 2 and 3 milliseconds, more packets can be transmitted on time and there is a plateau in the delay curve. After 3 milliseconds most packets can be processed on time, reaching a minimum delay with a skew of 6 milliseconds. When the skew exceeds 6 milliseconds the packets have all been processed and they sit idle in the output frame. The delay therefore grows linearly with this "excess" skew time.

The figure indicates a variation of 5.5 milliseconds in high priority traffic delay or (5.5/16.5) 33%. The effect on the low priority traffic is less noticeable, (2.5/20.5) 12%.

Figures 1.16, 1.17, and 1.18 present the results of similar experiments with decreasing average number of packets/frame: 3, 2, and 1 packet/frame respectively. It can be observed that as the number of packets/frame diminishes, the "optimal" skew value decreases. This is to be expected since the load has decreased and most packets will be processed in time for the next output frame.

The results do not seem to justify the increased complexity of a system in which the skews are predetermined as would be the case in a synchronous network. Remember that this is just the transmission delay for a single node. The lifetime of a packet inside a node is always greater that this value since the node must wait for an acknowledgement before reclaiming the buffer space, in which case the impact of the skew could very well be negligible.

Real Time Traffic

System Parameters:

Line Speed: 1.544 Megabits/sec (T1 carrier)
Frame Length: 10 Milliseconds
Number of Channels: 2 (Full duplex)
Number of Processors: 3 (PDP-11/20)

Frame Departure time: 0 milliseconds
Channel 0 Frame Arrival: 1 millisecond
Channel 1 Frame Arrival: 9 Millisecond

As in the Data Transmission experiments two different types of memory-memory move instructions were assumed. In the first type each individual word (16 bits) of real time information is transmitted individually in a software loop. The second type

assumes a slightly improved instruction set in which a factor of 4 speed-up is achieved by doing block transfers. In the experiment we considered two different slot sizes: small (80 bit) slots which could be suitable for voice communications and large slots (800 bits) which could be appropriate for facsimile or low quality image transmission.

The following table summarizes the experimental results. The numbers indicate the maximum number of slots that could be transmitted before the Real Time Transmission task fell behind the arrival rate and had to give up working on a real time packet.

	80bits/slot	800 bits/slot
1 word/move	31	15
4 words/move instruction	34	19*

* The limit here was dictated by the frame size. 20 slots of 800 bits (16000 bits) is more that the frame size (15440 bits).

The table shows that transmitting the maximum number of small slots using a single word/move instruction i.e. 31 slots, only uses 31*80=2480 bits of a frame or 2480/15440=16% of the frame capacity. Transmitting 34 small slots using a better move instruction only utilizes 34*80=2720 bits or 2720/15440=17.5% of the frame; a rather small improvement.

The transmission of large slots presents a different picture. Even when a single word/move instruction is used, the system can transmit 15*800 bits/frame or 12000/15440=78% of the frame capacity. Using the faster block transfer instruction indicates a 100% utilization.

The dramatic difference between the results indicates that the limiting parameter is the number of slots rather that the slot size. In other words, the overhead in

setting up the loop for transmitting a slot is the factor that dominates the switch capacity for real time transmission.

The following analysis determines this overhead component. The load on a processor, in the absence of other traffic types, can be modeled by: C' + N * (C + X) where C' is the overhead incurred by the processors in doing non-real time related tasks (inspecting queues, polling interrupts, collecting statistics, etc.) C+X describes the real time transmission loop. C is a constant overhead spent in loop control and X is the actual slot movement time (X is variable, depending on the type of move instruction used). N is the number of slots/frame.

The following equation describes the improvement in the switch capacity as a result of using the block transfer instruction on small slots: C' +31 * (C + X) = C' + 34 * (C + X / 4) from which we obtain, (after some algebraic simplifications): C = 7.5 * X i.e. the overhead of moving a real time slot is 7.5 times the cost of the actual movement. If we consider that a PDP-11/20 takes 5 microseconds to move a word (16 bits) from memory to memory then the time to move a small slot is:

X = 80bits * 1word/16bits * 5microseconds/word = 25microseconds.

The loop overhead time, C, is therefore 25 * 7.5 = 187.5 microseconds and the total time to move a slot is C + X = 187.5 + 25 = 212.5 microseconds. The time needed to transmit 31 slots is 31 * 212.5 = 6587.5 microseconds.

The latter figure indicates that it takes 66% of the time to transmit 16% of the frame (17.5% using the block move instruction).

The analysis of the improvement in the switch capacity when using large slots is described by: C' + 31 * (C + X) = C' +15 * (C + 10 * X) (C+10X is the time required to move slots that are 10 times longer).

The equation yields: C = 7.45 * X which is consistent with the previous result. The total time to move a large slot is: C + 10 * X = 17.45 * X = 17.45*25 = 436 microseconds.

The transmission of 15 large slots takes: 436 * 15 = 6550 microseconds. i.e. it takes 66% of the time to transmit 78% of the frame capacity.

The evidence above shows that the system is better utilized during the transmission of large size real time slots, and that any significant improvements will have to come from a reduced overhead. In other words, software table look-ups and loops can not perform a satisfactory job. Later in this section we will describe some ideas that should be considered in the implementation of firmware or hardware real time transmission schemes.

Mixed Real Time/Data Traffic

Figure 19 shows the effect of various levels of real time traffic on the average packet delay. Notice that for this particular experiment we were able to utilize 4 processors (PDP-11/20's) instead of the usual 3 processors as indicated in the previous experiment discussions. Moreover, the presence of the extra processors allowed us to process the real time traffic in a slighly different manner. For this experiments we divided the real time copy task into two subtasks/frame i.e. two processors could now co-operate in the disection of the real time packets for each input frame.

The figure shows that the real time load impacted the low priority packets more seriously than the high priority packets. Increasing the number of real time slots resulted in an increase in the average packet delay, as was to be expected.

The important result that can be observed from this experiment is the ratio

between an increase in the number of real time slots and the reduction in the average number of packets/frame that could be processed. If we observe the solid lines, representing the high priority traffic delays, we can estimate a ratio of 7.5 real time slots/packet (going from 10 to 40 real time slots resulted in a reduction in the number of packets/frame from 7 down to 3, i.e. (40-10)/(7-3)=7.5 slots/packet). If we ignore the slight variation in packet delay, the dotted lines (low priority traffic) associated with 32 and 40 real time slots indicate a decrease of 1 packet/frame (from 4 down to 3) that is to say, a ratio of 8 slots/packet.

1.3.3. Alternative Architectures: C.mmp Machine vs. Hydra Machine

The Hydra operating system currently in use on C.mmp creates a very different machine from the user's point of view than what is available on the bare hardware. Very early in the design stage, we had to decide whether to work under Hydra (i.e., on the Hydra machine), or on the bare C.mmp hardware. The reasons for the choice made are given here, after some review of the basic features of the two machines.

Design Features of the C.mmp Machine

The most important feature of C.mmp, of course, is the set of 16 independent, asynchronous processors, sharing a common main memory. In addition, there are a total of 16 different memory ports, allowing up to 1024K words each. The processors and ports are joined by a 16 x 16 crosspoint switch, allowing a maximum of 16 simultaneous processor-memory conversations. In addition to the shared memory, each processor has 4K words of local memory accessible.

Because of the limitations of the 16 bit address, not all of memory is directly addressable by a processor at any one time. Memory is organized into pages of 4K

words each (8K bytes). A set of relocation registers connected to each processor allows a maximum of 8 pages to be addressed at once, for a total of 64K words. This small address space has an enormous effect on the construction of large programs on C.mmp, since they must be organized as segments of 4K words each. Also the user must supply his own convention for segmentation, overlays, and the like.

Each processor has the ability to start, stop, or interrupt any or all of the other processors by means of interprocessor interrupts (IPIs). These IPIs can occur at different priority levels, and allow fast interprocessor communication. There are also four levels of operation priveleges: the 00-space, 01-, 10-, and 11-spaces. The 00-space has the lowest priveleges (i.e., user-space), while the 11-space has the highest (e.g., operating system). Each space has its own set of 8 relocation registers which are independent of the relocation registers in any of the other spaces.

The I/O devices are attached to thee Unibusses of specific processors. This means that any I/O interrupts can only come to the processor which owns the device (hence the need for fast interprocessor communications). There is, however, one "device" common to all the processors: a 60-bit global clock, which has a resolution of 4 microseconds. In addition, each processor has an interval timer clock which is driven off this global clock, with resolution of 16 microseconds.

Finally, C.mmp allows the use of microcoded PDP-11/40Es on the system. Eventually, a maximum of twelve 40Es will be included in the system, with the other four processors being PDP-11/20s. Thus, it will be possible to implement special-purpose instructions for particular tasks by use of the writable microstore.

Current Status of C.mmp Machine

Currently there a total of nine processors connected to C.mmp, five PDP-11/20s

and four PDP-11/40Es. However, the integration of the 40E processors into C.mmp is not yet complete. Some problems have been discovered in the original interface design which usually keep them from performing reliably on C.mmp. Hence, the model 40s are usually partitioned out of the running C.mmp. The total number of running processors is usually around five (all the 11/20s), though the number has occasionally been as low as three, and as high as seven or eight.

Currently, there are a total of 680K words of memory on 16 ports. Occasional parity errors or bad ports are detected, but the memory, by and large, is reliable. The rest of the hardware of C.mmp (IPIs, global 60-bit clock, interval timers, 00-11 spaces, relocation registers, and switch concurrency) are also operating relatively reliably.

Design Features of Hydra Machine

Hydra is the operating system designed for C.mmp. Its philosophy is to create a general-purpose machine out of C.mmp in a timesharing environment. Hydra hides many of the details of the underlying machine that are unnecessary to know about from a user's point of view. The operating system also provides the tools for creating a highly protected system based on the use of capability lists, access rights on capabilities, etc.

Hydra is divided into two parts: the subsystems and the Kernal. The Hydra Kernal is a set of routines which cover up all the "dangerous" aspects of the machine. These routines are not the operating system, but rather the tools with which to build the operating system. The Kernal operates in the privileged 11-space; users operate in the unprivileged 00-space. Certain "dangerous" instructions, and all interrupts, are handled only in the Kernal.

Outside the Kernal are components called "subsystems", which implement all the

necessary functions of a timesharing operating system: a file system, command language, scheduler, compilers, utilities, resource allocater, etc. These subsystems do not have access to the underlying machine -- only the Kernal has that capability. In a sense, they are all user programs, and in principle, several disjoint sets of subsystems could be operating concurrently. Hence, "Hydra" is a generic name applied to the Kernal and some set of subsystems (however, there exists only one set currently).

Many more "processes" are allowed to exist than processors. The Policy Module subsystem (by using tools in the Kernal) allocate processors to processes via a specific scheduling policy. The user cannot guarantee when a process will get scheduled, or how long it retains control of the processor -- this is a result of Hydra being a timesharing machine.

Pages for a user process are allocated by the Kernal -- the user has no control over which physical pages are allocated to him, or more importantly, no control over which pages go into which ports. This means that he has no control over the amount of memory contention that takes place when several processors are all making continuous memory requests. They will all run significantly faster when the memory requests are to different ports, thus utilizing the concurrency of the crosspoint switch. A high memory contention rate causes degradation of processor speed.

DCA Communications Task on the C.mmp Machine

There are a number of strong advantages for using the C.mmp machine for the communications task. Basically, it is the most appropriate machine for running a real-time program on. One has complete access to the global clock, interval timers, and IPIs. Precise timing is possible because interrupts are fielded directly -- hence polling is unnecessary. In addition, pages can be allocated specifically to minimize memory

contention and thus increase processor performance. User microcode can be easily inserted for special-purpose instructions. In short, this is just the advantage of being able to get to the all the hardware for maximum performance and precision.

However, there are some major practical disadvantages in using the bare C.mmp machine. Specifically, there is virtually no support available for running on a stand-alone C.mmp. This means that the user would have to write his own software for handling all interrupts (I/O, IPIs, hardware errors, etc.), drivers and error-handlers for all desired I/O devices, recovery procedures from exceptional conditions, and so on. In essence, this means writing a mini-operating system.

In addition to writing a great deal of low-level code, one must also worry about debugging such notoriously difficult-to-debug code as interrupt handlers. Complicating the situation further is the relatively small amount of time each day when one can sign up for exclusive use of C.mmp -- a small point, but one which could have a great effect on the length of time before a body of software is up and running.

DCA Communications Task on the Hydra Machine

The main advantages of running under Hydra are exactly those features without which it would be very difficult to run: a file system, process starting/stopping, availability of I/O support, exceptional condition handling, interrupt handlers, debugging aids. There is also a support group for Hydra-related functions which are available for fixes or improvements in the operating system. Finally, there is the advantage that most of the debugging can be carried out in a timesharing mode with other users, hence the availability of the machine is greatly increased.

However, there are some major disadvantages in using the Hydra machine for the communications task. All interrupts are intercepted by the Kernal -- the user is

never allowed to field an interrupt directly (which is a distinct disadvantage for a real-time program). The best possible compromise is a form of polling, where the Kernal interrupt-handler sets particular bits for specific interrupts.

Hydra allows lots of high-level abstractions: processes, semaphores, ports and messages, capabilities, etc. However, the use of these abstractions is very costly in processor time (e.g., a context swap, where a processor is switched from one process to another, takes about 20 ms. on a PDP-11/20). Hence, the use of these natural, but expensive, abstractions is prohibited by the real-time nature of the program.

Another problem is that Hydra insists on doing things underneath any user program. For example, time slice timeouts, I/O interrupts, and periodic scheduling interrupts (to allow a processor to swap to a higher priority process) are all performed invisibly to a user program. However, they all take processor time in potentially deadline situations (such as getting Class I traffic into an output frame in time). Several things were done to reduce or eliminate these problems: (1) Runs were done on a dedicated system with infinite time slices -- no other users were allowed to run at the same time; (2) specific sections of the operating system were patched to eliminate periodic low-level activity; and (3) the system was designed so that no I/O activity took place during the actual running of the simulation (though it was allowed during input and analysis).

Conclusions

For overwhelmingly practical reasons, the Hydra machine was selected to be used. Looking back, it is doubtful that we would even be close to having a running system now if the C.mmp machine were chosen. Much low-level code concerned with the smallest details of the PDP-11s would have to have been written and debugged --

and debugging of time-dependent interrupt routines can be tedious and time-consuming, even for an expert.

Yet the choice of the Hydra machine, while pragmatic, had many bad effects. We tried to make use of Hydra in a way never intended by its designers (dedicated machine, patching out annoying interrupts, allowing infinite time-slices, etc.). Moreover, the choice dictated a fundamental part of the design: interrupts had to be polled rather than handled directly. Polling has other advantages, such as not having to save state, but we were barred from even considering precise handling of interrupts. Hydra also has a strong effect on the time required to load a relocation-register: the basic machine instructions require 10-20 microseconds, while the corresponding Hydra function requires 300 microseconds. We have been able to largely avoid the address space-problem by structuring the page-configurations so that dynamic relocation-register loading is not necessary. However, keeping this structure may not be possible if the system size increases substantially. In this case, dynamic relocation-register loading would probably be the required solution -- one that would be much more expensive under Hydra than on the bare C.mmp machine.

1.3.4. Alternative Design for Class I Traffic Transmission

Class I traffic (real-time) is essentially a "scatter-write" operation. That is, Class I slots arrive contiguously in an incomimg frame at a node, but are transmitted out on other channels in slot positions independent of their incomimg frame slot position. (Alternatively, the operation can be viewed as a "gather-read", where the contiguous slots in each outgoing frame are gathered from various input frames and slot positions.) The Class I slots may of course be of varying sizes, but they are assumed

to contain an integral number of characters and to remain fixed in size for the duration of a connection.

The current implementation for the "scatter-write" operation is by software. An input frame is received and placed in contiguous memory by the input Line Interface Device (LID). Slots are then copied by software to appropriate buffer positions to create contiguous output slots. (Note that copying was performed, rather than simply passing a pointer. The reason for this is that the output LID was already assumed to be following a two-level structure of pointers. Passing slot pointers would necessitate assuming a hardware LID design involving three levels of pointer nesting: a pointer to the Channel Command Vector (CCV); pointers in the CCV to packets; pointers in the Class I packet to individual slots. This seemed like too much to assume for a hardware device.)

The results of simulation runs for Class I traffic indicate that Class I transmission by software is an expensive proposition. Moreover, block transfers or even passing pointers would not gain very much because of the relatively large amount of overhead per slot, especially small slots, spent in table-lookups, queue operations, etc. Since the Class I transmission is basically a straightforward operation, one would like to have the software program of a node simply simulate the role of a telephone operator, setting up and breaking down connections as needed, but not involved with the actual transfer of information along the connections.

One approach which satisfies this condition would use a special-purpose microprocessor connected directly to the input line and able to access the output buffers in memory. This microprocessor would refer to a table to obtain the destination buffer address and slot length for each incomimg Class I slot. Thus, as the

Class I packet begins to arrive, the microprocessor would read the first entry in the table to determine starting address and length, and then store the corresponding number of incomimg bytes into sequential locations starting with the stated address. Then the table would be referenced again for the destination and length of the second slot, and so on. Thus, instead of depositing incomimg Class I slots into contiguous locations in memory (i.e., an "input buffer", which is the assumption in the current simulation program), this microprocessor would be constantly referencing the table in memory and storing the incomimg slots in their appropriate positions in the output buffers. The net effect of such an operation, with a microprocessor for each incomimg channel, is the creation of contiguous output Class I buffers, ready for sending out on the next link.

The primary task for the node software here is updating the table as changes occur in the Class I packet composition (e.g., normal setting up and breaking down of connections). The situation is complicated slightly by the need for a ring of three buffers for Class I traffic -- to ensure that an incomimg frame is not switched to a different buffer in mid-stream (see Appendix I) -- but the added complexity is relatively minor.

The operation of several microprocessors (one for each input channel) accessing main memory continuously could become a major bottleneck. For example, assume that each microprocessor obtains a 16-bit word from the LID; then about 10.4 microseconds elapse between words (at the rate of a T1 carrier, 15440 Kbps). During each 10.4 microsec. interval, the microprocessor will, in the worst case, have to perform a read access on the slot-destination table, and a memory-write of the incomimg word to the specified destination. Of course, most slots would be more than 1 word long, so the

table-read would only have to be done intermittently. But a worst-case analysis would have each of the bnn microprocessors performing a memory read and write cycle every 10.4 microsec. In addition, the output LID for each channel would be reading a buffer at the rate of one read access every 10.4 microsec. Hence, we see that

$$n (2R + W) \leq 10.4$$

where

n is the number of channels
R is the memory read-access time
W is the write cycle time.

For example, if R=0.5 microsec. and W=1.0 microsec., then in the worst case no more than five T1 lines could be handled by the memory. Note that this also assumes high priority for the microprocessor memory accesses, implying that any additional source of memory accesses (e.g., instruction fetching) would be degraded. Interleaving memory or providing multiple ports could alleviate this possible bottleneck.

1.4. Appendix I : Class I Traffic, Further Considerations

There are some further issues and problems that must be dealt with before the Slotted Envelope design can be completely defined. This section describes some special problems associated with the handling of Class I traffic and suggests some possible approaches to resolving them.

1.4.1. Class I Reservation Protocol

The maintenance of routing tables is one of the outer loop functions that will be studied during the second year of the contract. in this section we will present some preliminary thoughts on the matter.

Reservation of a Class I channel across the network would be done in several stages. First, upon call-initiation, reservation-request messages would be sent to determine a feasible path through the network for the connection. Nodes in the path would reserve a slot in the Class I portion of their frame for the call, although the slot would not actually be allocated it until the call-initiation was successfully completed (i.e., the called party answered the phone). At this point, the reserved slots would be allocated, on a pairwise basis on the nodes along the path of the call. The Class I reservation tables used by the nodes are depicted in Figure A.1.

Since nodes at both source and destination ends of a specific link would have to agree in advance on changes to the Class I portion of frames traversing that link, some sort of slot allocation/deallocation protocol would be required which would enable agreement on <u>when</u> a Class I change is implemented as well as <u>what</u> the change is. The addition of a time requirement complicates the situation since delays or line errors

could occur, forcing timeout and retransmission of the CCIS packets in the presence of a potential deadline.

Two basic strategies in the protocol are possible. The first, which could be termed the time relative approach, is to have the CCIS always reference a change-time (frame number) that is a constant time in the future from the frame containing the "request for allocation" (ALLOC) packet. The responding node would acknowledge the ALLOC packet with an acknowledgement (ACK) packet containing the same information. Retransmission of the ALLOC packet (with a new proposed change-time) would occur if the ACK information did not match, or no ACK arrived before timeout. The advantage of this approach is that a deadline situation is avoided, since any retransmission would put off the change-time by an amount equal to the delay encountered. The danger with this approach is that an indefinite postponement of the change could occur, as illustrated in Figure A.2(a). Node A originates the ALLOC packet in frame 1, with a change-time of +7 relative to 1, or 8. The ACK packet from node B is delayed, so A times out and resends the ALLOC packet in frame 6 with a new change-time of 13 (6+7). This ALLOC message crosses with the delayed ACK from B. Node A must now send another ALLOC packet since the ACK information does not match, while B must also respond to the new ALLOC packet. An "out-of-phase" situation like this could continue indefinitely and cause an unpredictable delay in establishing a Class I change between two nodes.

The second approach, termed time absolute, would have retransmissions reference the same future time as originally proposed in the first ALLOC/DEALLOC CCIS packet. No "out-of-phase" situation is now possible, but the new danger is that of only partially completing the handshaking procedure before a deadline expires. This is illustrated in Figure A.2(b).

In both these strategies, the chance of error can be reduced by appropriate selection of timeout durations, change-time increments, and CCIS packet priorities. Improvements in these strategies are obviously also possible, though it is difficult to ensure that any protocol is absolutely foolproof against all possible situations.

1.4.2. Class I Slot Slippage and Overwrite

As mentioned before, in Section 2.2, there are some problems that might arise during the transmision of Class I traffic. These problems are best illustrated by an example. The following two paragraphs first explain the figure and then illustrate the slippage/overwrite effect.

Figure A.3 schematically shows the flow of information through a single input channel (IN) and a single output channel (OUT). Successive input frames are labelled (IN1, IN2, etc.), as are the output frames. For clarity, the Class II portion of each frame has been omitted, and only four Class I slots (a, b, c, and d) are shown in a frame. To simplify the explanation, it is assumed that all four incoming Class I channels all go out on the same output channel. In general, frames on input and output channels will not be synchronized. In this example, the output frame lags the input frame by a duration of β (=t2-t1). Finally, the two buffers of the OUT channel are explicitly shown: during frame OUT1, for example, buffer 1 is being filled, while buffer 2 is being transmitted and is locked out (indicated by the heavy dashed line).

At time t1 in Figure A.3, incoming frame IN1 has been completely read in and is made available for processing. The switch program then has until t2 to transfer slots into buffer 1 of OUT; at t2, the buffers are switched and any subsequent transferring must go into buffer 2. In the example, assume that the switch processor's speed and the lag duration β are such that the switch program can:

(1) always transfer slots a and b,
(2) usually transfer slot c, and
(3) never transfer slot d,

during the interval β before the buffers are switched (at t2, t4, etc.). If the frame duration is denoted by F, and the frame lag by β, then the delay through the node for Class I channels a and b is $F+\beta$. The delay for channel d, which is always delayed an extra frame, is $2F+\beta$. Channel c, however, can experience either of these delays, depending on whether each slot c_i gets transferred into the earlier buffer or later buffer. At the transitions between the two delays for channel c, there occurs either a slip (if the delay is increasing), or an overwrite (if the delay is decreasing). Figure A.3 shows a slip in channel c during the OUT2 frame. Additionally, an overwrite would have occurred in OUT3, if c_3 had been transferred during the (t5,t6) interval.

While occasional slips or overwrites might not have any strongly noticeable effect on voice conversations, other real-time connections, such as encrypted voice, can be seriously affected if an error occurs when an encoding key is being transmitted.

It should be noted that the specific effect described occurs because of variations in response time to a frame arrival. This is certainly more likely in a software implementation of "scatter-write" (e.g., a frame-arrival interrupt may not be taken immediately because of another higher priority interrupt); a hardware realization would be much less susceptible to these variations, though it would still depend on the source node to time the frames precisely.

Even if all timing and response times are precise, an overwrite could still occur in a perfectly innocent way: If slots b and c were deleted after input frame IN1 (due to normal disconnection), then frame IN2 would only have two active slots to transfer, a

and d, which could be done in the interval (t3,t4). Hence, d's delay would decrease from $2F+\beta$ to $F+\beta$, causing an overwrite in frame OUT2.

A solution to these problems is simply to use an extra buffer, so that an input frame is never switched to a new buffer in mid-stream. The penalty for this is that the cross-node delay for each Class I channel is now $2F+\beta$ units. (A hardware implementation, by doing the transfer as the data is arriving, could reduce this to $F+\beta$.) Furthermore, buffer selection would be decided by the arrival time of an input frame; if the input channel and output channel were almost synchronized, any relative drift between the clocks at the two nodes could cause a discontinuity in buffer selection (compared to the normal cyclic order), and hence a slip or an overwrite for all the Class I channels on that input frame. This is expected to occur rarely, however, especially if the relative drifts between nodal clocks is small.

1.5. Appendix II : Script Generation Parameters

The following (Figures A.4, A.5, A.6) is an example of the dialogue with the Script Driver for specifying the node and traffic characteristics.

- For those parameters having relatively constant values, defaults are built in. These default values appear enclosed in angular brackets after the question.

- Run Number is a number used for identifying the experiment and associating the statistics printout with the parameters specified by the user.

- The run durations to obtain steady state and after steady state are specified in number of frames sent on any single line.

- If two or more channels are identical, only one need be specified. The Script Driver will duplicate the channel specifications for additional instances of a previously specified channel.

- The distributions for packet types are given as percentages and their interpretation is explained in Section 3.1.

- The average number of packets per channel is A*T where T is the frame duration and A is the mean arrival rate of a Poisson process.

- Packet lengths are specified in number of bytes. The length is drawn from a uniform distribution limited by the maximum and minimun values specified by the user.

- The distribution of packet destinations indicates the average fraction of packets that go to each output channel.

- The real time packet composition is indicated explicitly so that a desired pattern can be specified.

- The frame departure and arrival times are relative times. Thus one of the event occurs at time=0 and other events occur at the indicated relative times.

The entries are displayed in a coded format. This is to enable the user to identify a parameter by its code name.

The codes are:

R Run number

S	Seeds for random number generator
B	number of frames Before the steady state is obtained
H	How long the system is to run
Nx	specification for channel x
Tx	speed of channel x
Lx	maximum amd minimum packet length (in bytes) on channel x
Px	packet type percentages on channel x
Dx	packet Destinations percentages from channel x to 0,1,2 3
V	Composition of the real time packets.
I	events vector i.e relative times for varios events
	event definitions:
0	frame arrival on Channel 0
1	frame arrival on Channel 1
2	frame arrival on Channel 2
3	frame arrival on Channel 3
4	frame departures
5	Statistics Collection

In response to Which line do you want to change, typing the code letters would cause the corresponding questions to be repeated.

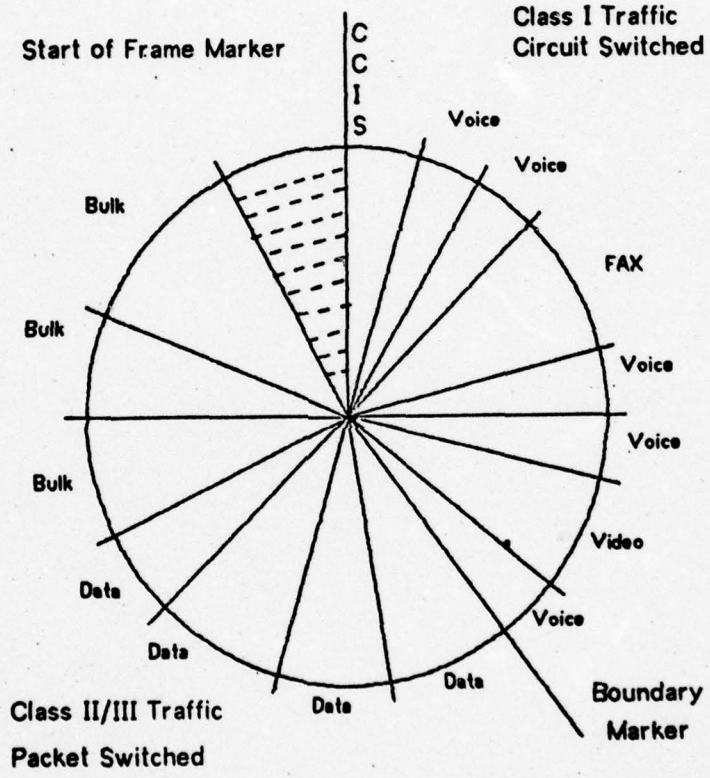
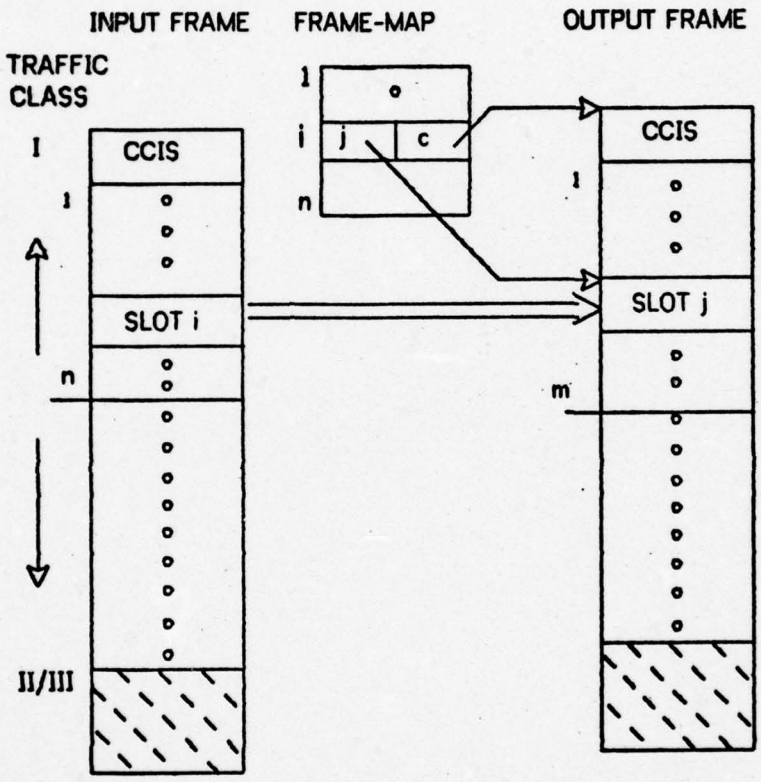
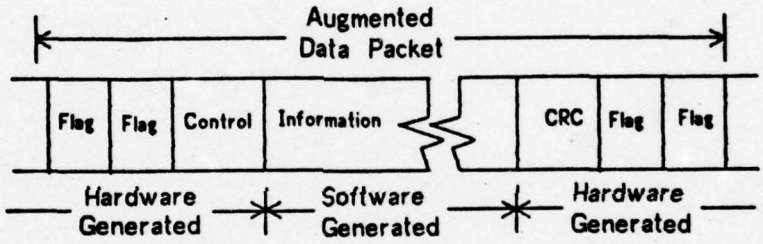


Figure 1.1 - An Integrated Network Frame



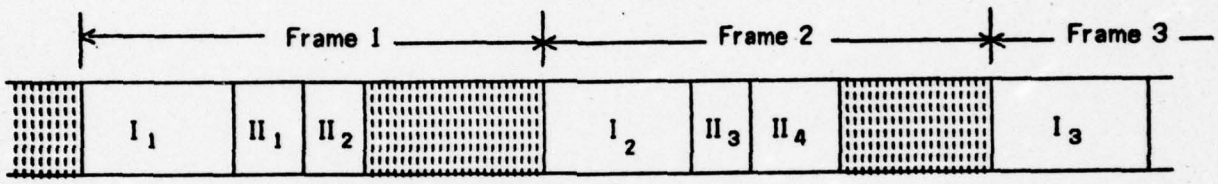
Processing of Class I Traffic

Figure 1.2



Packet Transmission Format

Figure 1.3



← Direction of Flow

I_i = Class I Packets

II_j = Class II Packets

Shaded area = Idle characters

Snapshot of Frame Transmission

Figure 1.4

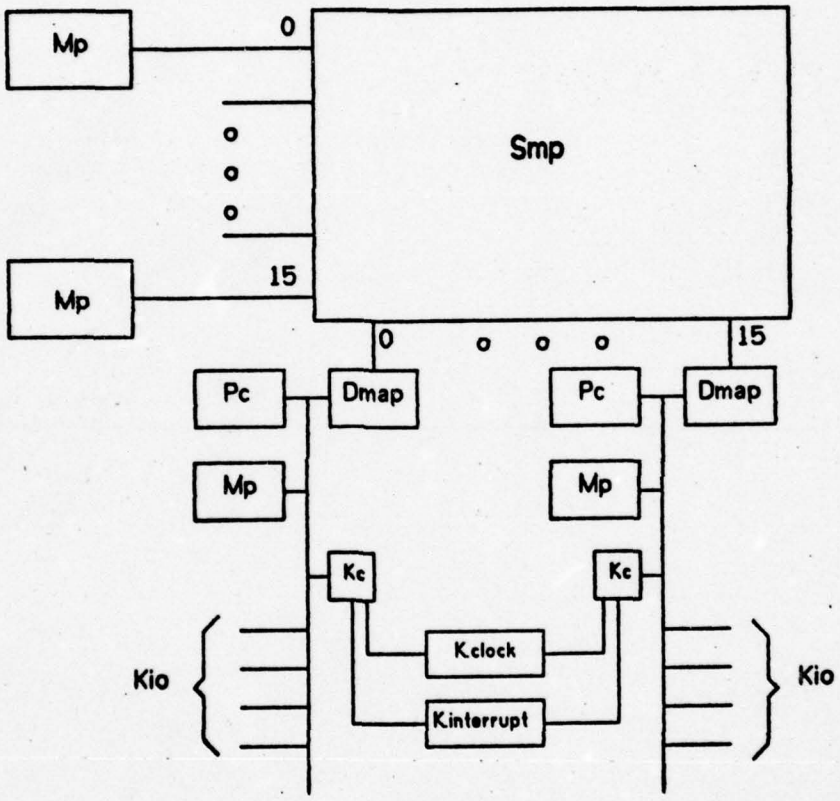


Figure 1.5 - C.mmp

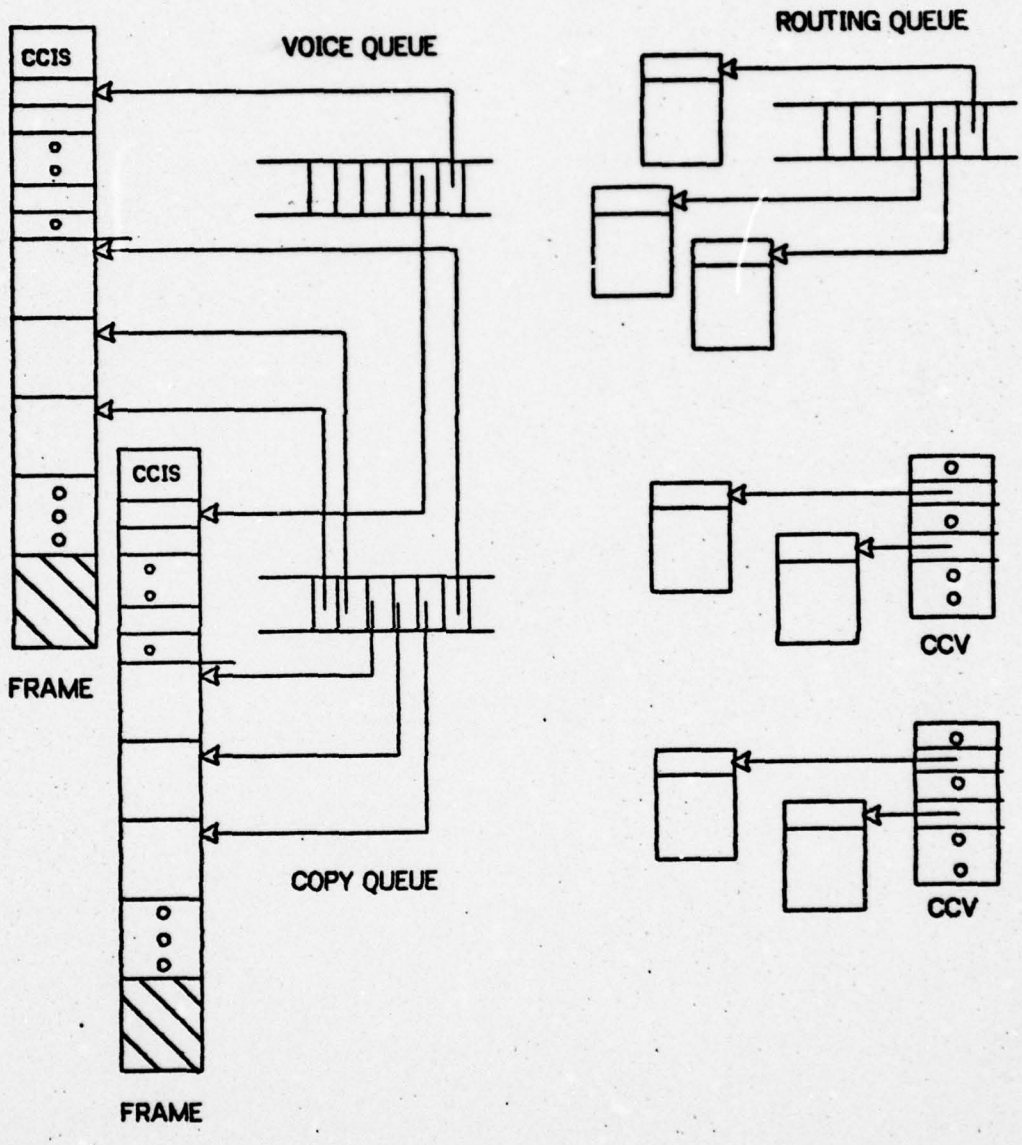


Figure 1.6 - Task Queues Organization

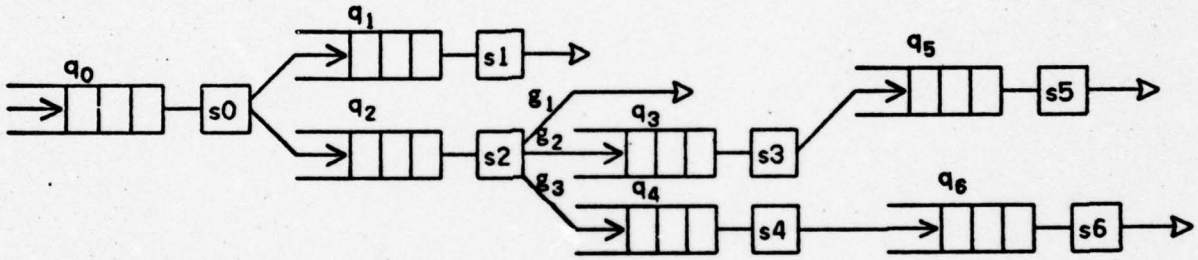
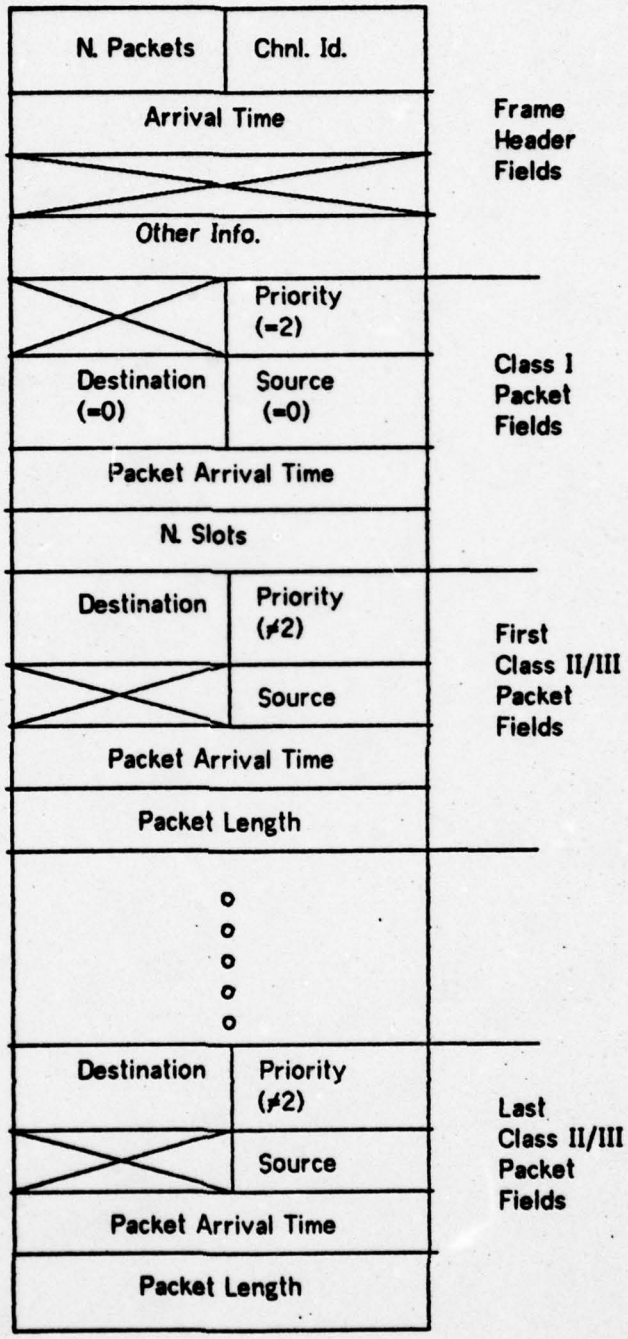


Figure 1.7 - Task Sequence Organization



Input Frame Structure (Script Driver)

Figure 1.8

```
                                                    Date of run:     14-Jun-76 21:
Statistics for Run number : 1001
PACKET DELAY AS A FCN OF REAL TIME LOAD AND NUMBER OF PACKETS/FRAME

Input Parameters


S  16293       0
S    100           H   1000          F   10          C   2

N0   Specifications for Channel  0
T0   1544          A0    1          L0    282    26
P0       0   20    0   80          D0    0  100

N1   Specifications for Channel  1
T1   1544          A1    1          L1    282    26
P1       0   20    0   80          D1  100    0

V   Composition of Real Time Slot
 Num From To Bytes
 10    0   1   10
 10    1   0   10

I       4   0       0   1       5   5       1   9

N  NumVCTask   EnabSTcomp   EnabVmove   EnabDmove   EnabBtfr   StatCollection
     2            0           1           1           1          Data


Number of frames sent/channel in steady state :      1000.

Channel      Total Pkts Sent      Average Pkts/frame to Channel  0   to Channel  1
   0              678.                0.68              0.000            1.000
   1              693.                0.69              1.000            0.000

  On       Total Bytes       Average     Data Cap.     Total Cap.    Total  Cap.
Channel      Sent            Byt/Pkt      Used        Data  Used    ClassI Used
   0       103569.000        152.757      0.057         0.054          0.052
   1       103945.000        149.993      0.057         0.054          0.052

Fraction of Priority Classes

Channel   Data High   Data  Low     Bulk High   Bulk  Low
   0        0.000       0.204        0.000       0.796
   1        0.000       0.199        0.000       0.801

Number of invalid frames :      0
```

Analysis Output

Figure 1.9

```
Noverrun      =    0   (# frames Script Generator was late for)
Vstattooslow  =    0   (# times Analysis process missed Class I pkts)
DPoverrun     =    0   (# times DPtask overflowed CCVs)
VCoverrun     =    0   (# times VCtask was too slow)
```

REALTIME DELAYS

Fraction/Number of slots transferred in n frames:

Source Chan	Skew (ms)	n= 0	1	2	3	4	5	6	7+
0	5	0.0000 0	0.0000 0	1.0000 2500	0.0000 0	0.0000 0	0.0000 0	0.0000 0	0.0000 0
1	5	0.0000 0	0.0000 0	1.0000 2500	0.0000 0	0.0000 0	0.0000 0	0.0000 0	0.0000 0

DATA PACKET DELAYS

		DELAY (ms)				SPACE*TIME (byte-ms)	
Priority	# Pkts	Mean	Std Dev	Min	Max	Mean	Std Dev
Control	0						
Datahi	345	22.86	4.111	15	25	288.4	155.9
Datalow	695	23.37	3.693	15	25	280.8	151.7
Bulkhi	1047	24.00	3.917	15	45	304.1	153.8
Bulklow	1366	26.53	6.107	15	55	324.7	194.0
Totals	3453	24.76	5.098	15	55	306.0	171.4

Analysis Output

Figure 1.10

TASKS CALLED AND PERFORMED BY PROCESSES

		VCOPY	XADR	DCCPY	DPROS
Process 2:	Tasks Called =	16462	16462	-32612	16462
	Tasks Performed =	1039	224	346	370
	Ratio =	0.063	0.014	-0.011	0.022
Process 3:	Tasks Called =	17394	17394	-30748	17394
	Tasks Performed =	1104	263	394	371
	Ratio =	0.063	0.015	-0.013	0.021
Process 4:	Tasks Called =	16758	16758	-32020	16758
	Tasks Performed =	1165	249	356	365
	Ratio =	0.070	0.015	-0.011	0.022
Process 5:	Tasks Called =	17217	17217	-31102	17217
	Tasks Performed =	1090	253	412	402
	Ratio =	0.063	0.015	-0.013	0.023

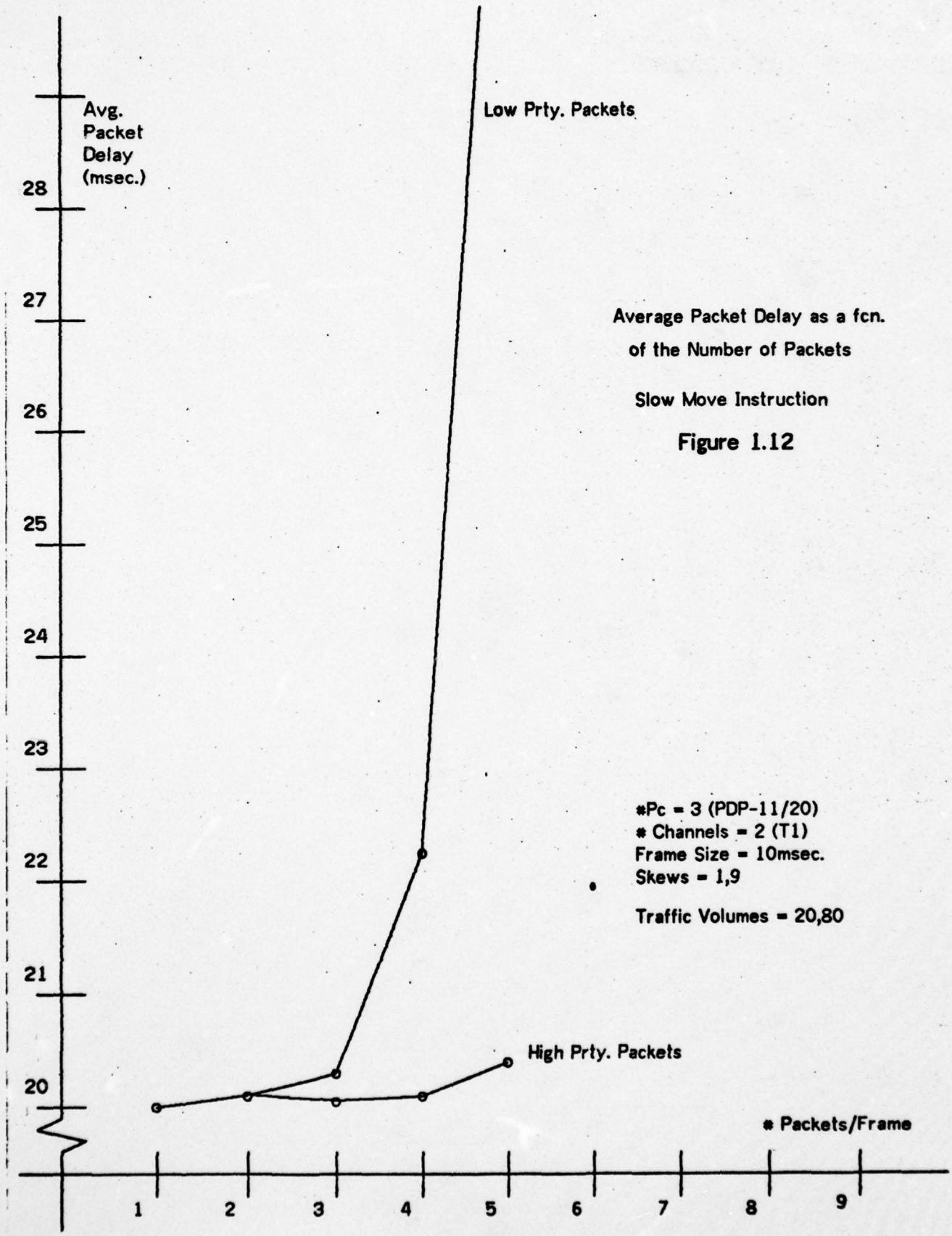
AVERAGE QUEUE LENGTHS

Queue	Avg Lngth	Nenter	Nlock	Nspin	Lock/Enter	Spin/Lock
Avail	798.634	19005	1291	1509	0.068	1.169
QVcopy	0.000	10474	2750	4795	0.263	1.744
QXaar	0.000	2350	306	306	0.130	1.000
Q1Dcopy	0.000	643	44	50	0.068	1.136
Q2Dcopy	0.000	2523	152	166	0.060	1.092
Q1Dpros	0.003	681	77	95	0.113	1.234
Q2Dpros	0.022	2732	361	474	0.132	1.313
Freebuff	798.634	3015	64	66	0.021	1.031

Samples =1001

Analysis Output

Figure 1.11



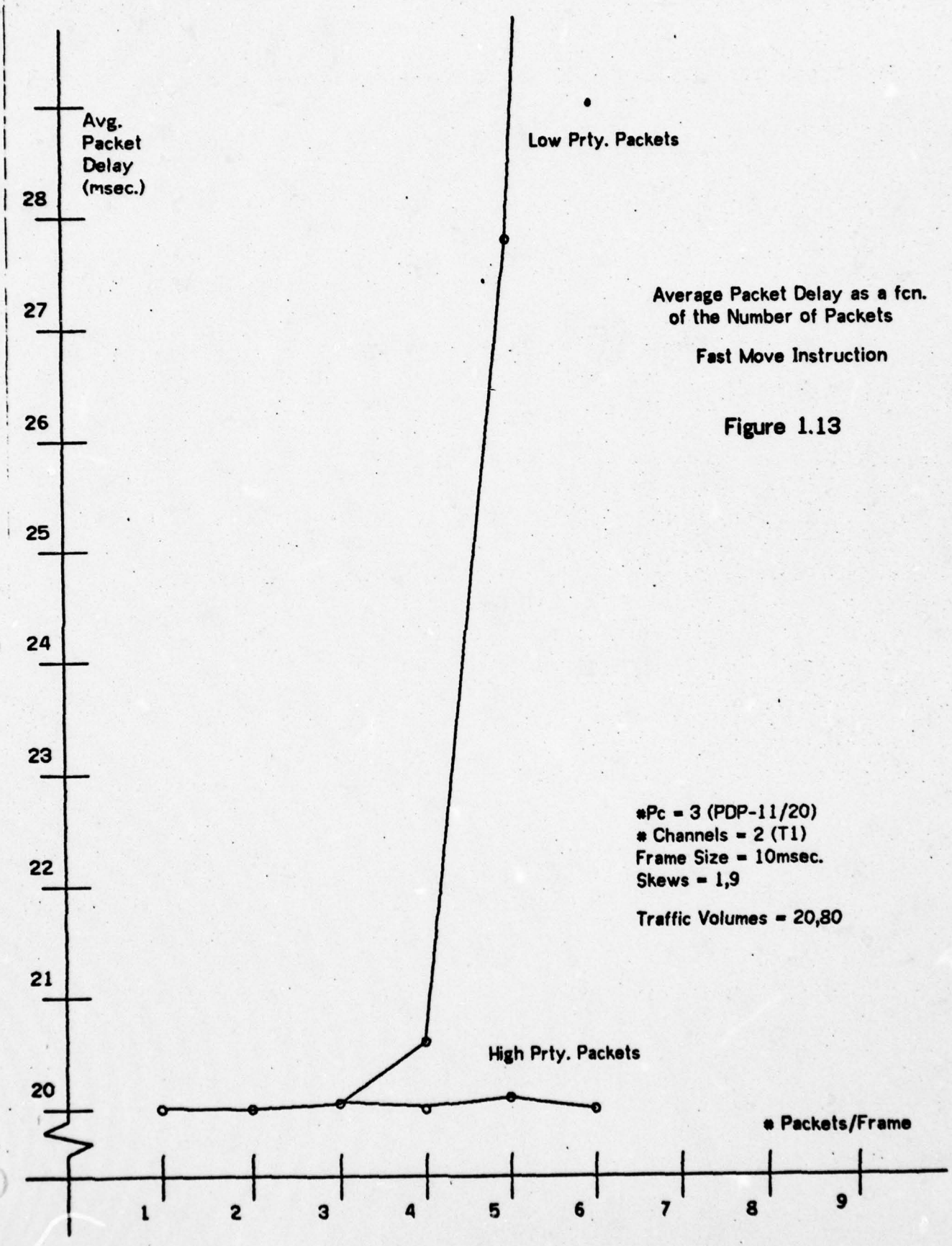
Average Packet Delay as a fcn.
of the Number of Packets

Slow Move Instruction

Figure 1.12

#Pc = 3 (PDP-11/20)
Channels = 2 (T1)
Frame Size = 10msec.
Skews = 1,9

Traffic Volumes = 20,80



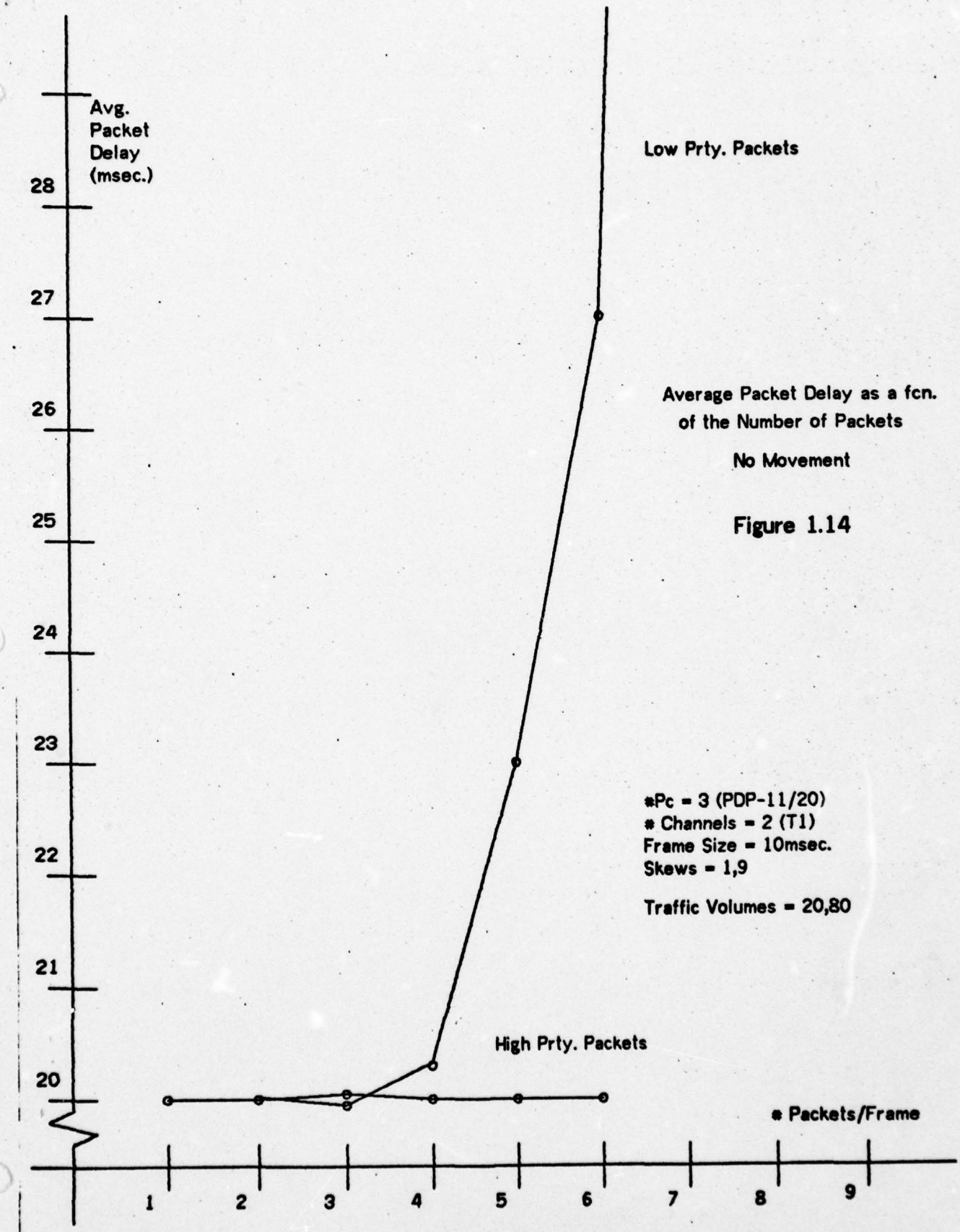
Average Packet Delay as a fcn. of the Number of Packets

Fast Move Instruction

Figure 1.13

*Pc = 3 (PDP-11/20)
* Channels = 2 (T1)
Frame Size = 10msec.
Skews = 1,9

Traffic Volumes = 20,80



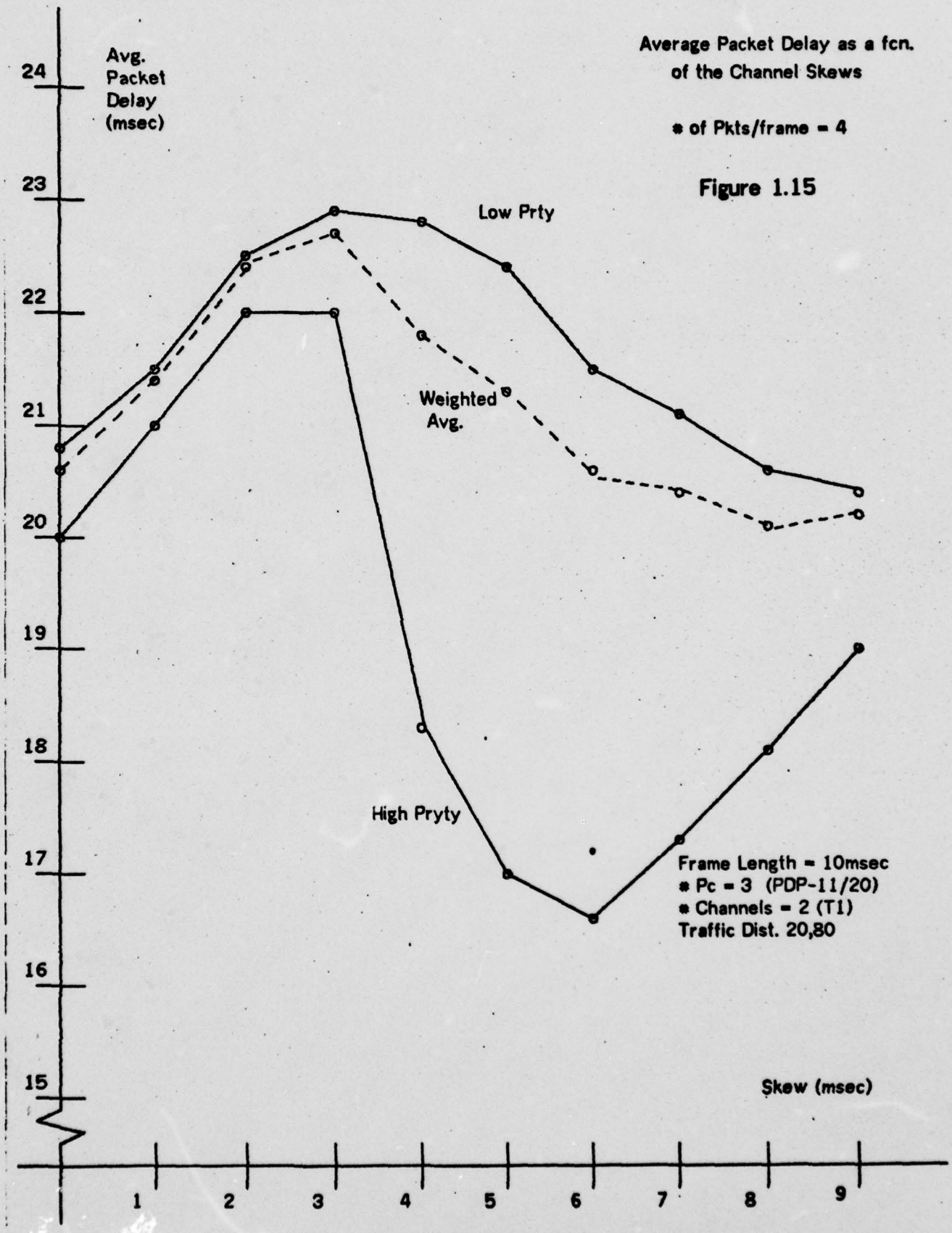
Average Packet Delay as a fcn.
of the Number of Packets

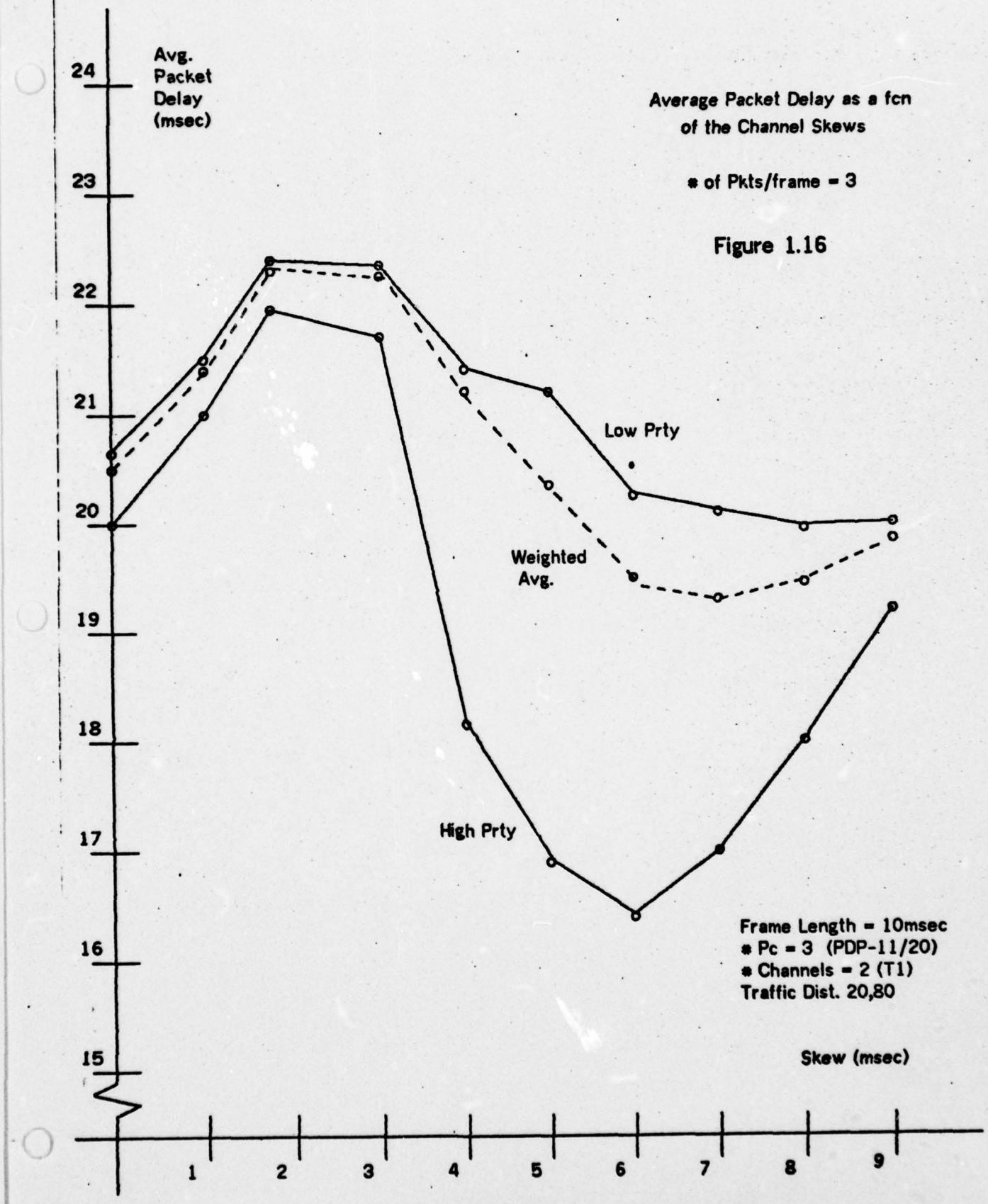
No Movement

Figure 1.14

#Pc = 3 (PDP-11/20)
Channels = 2 (T1)
Frame Size = 10msec.
Skews = 1,9

Traffic Volumes = 20,80

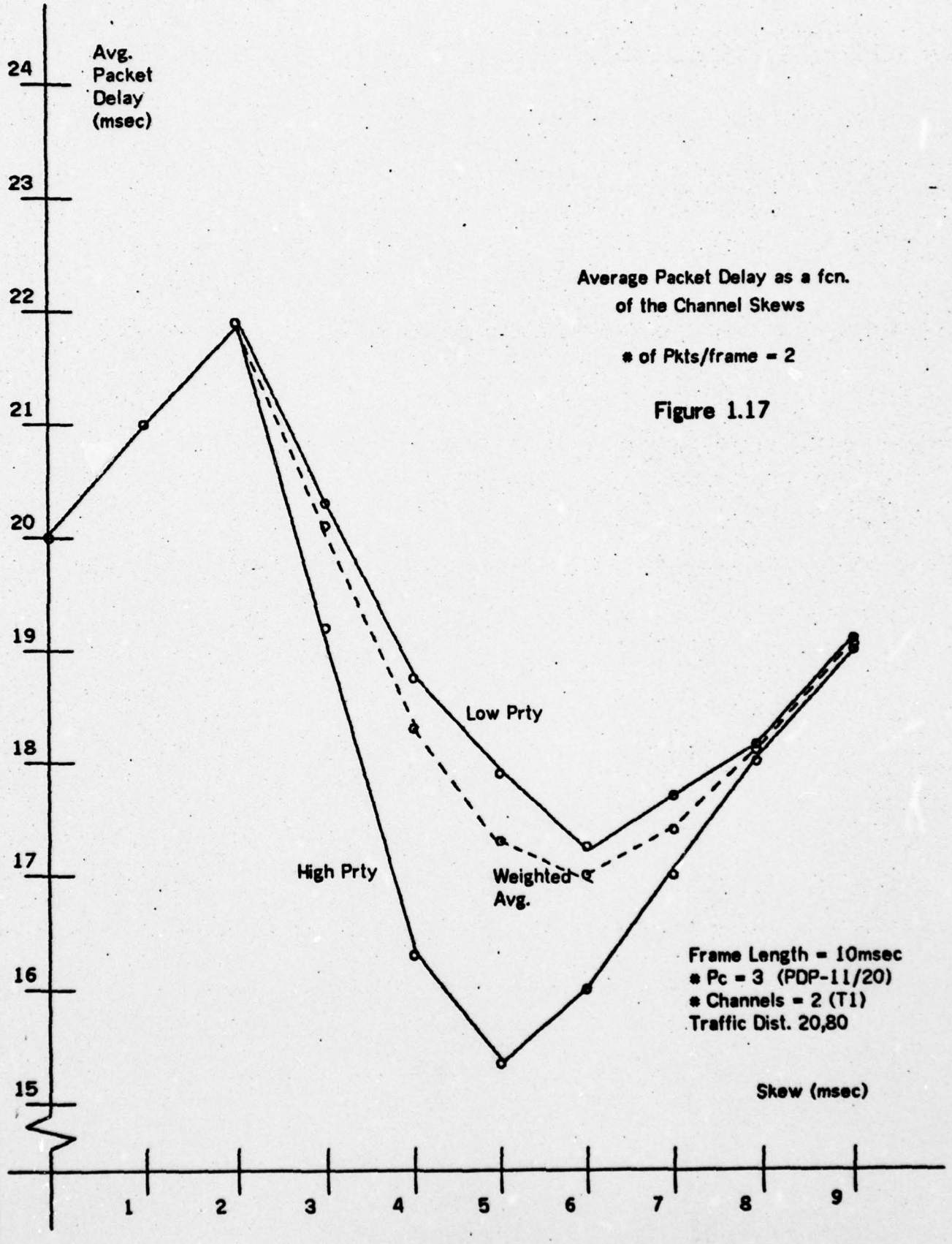

Average Packet Delay as a fcn.
of the Channel Skews
# of Pkts/frame = 4
Figure 1.15
Avg.
Packet
Delay
(msec)
24
23
22
21
20
19
18
17
16
15
Low Prty
Weighted
Avg.
High Pryty
Frame Length = 10msec
Pc = 3 (PDP-11/20)
Channels = 2 (T1)
Traffic Dist. 20,80
Skew (msec)
1
2
3
4
5
6
7
8
9



Average Packet Delay as a fcn of the Channel Skews

# of Pkts/frame = 3

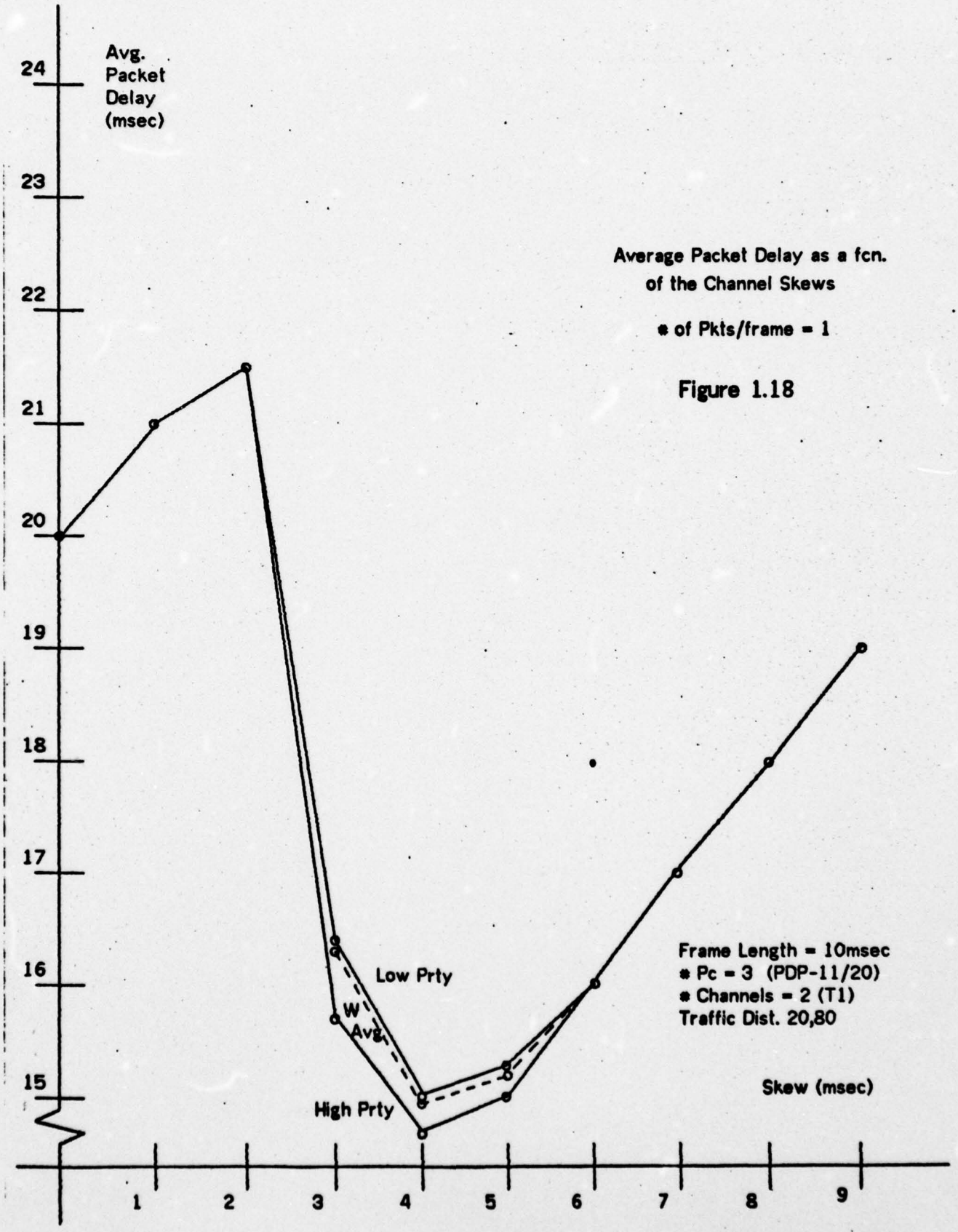
Figure 1.16



Average Packet Delay as a fcn.
of the Channel Skews

# of Pkts/frame = 2

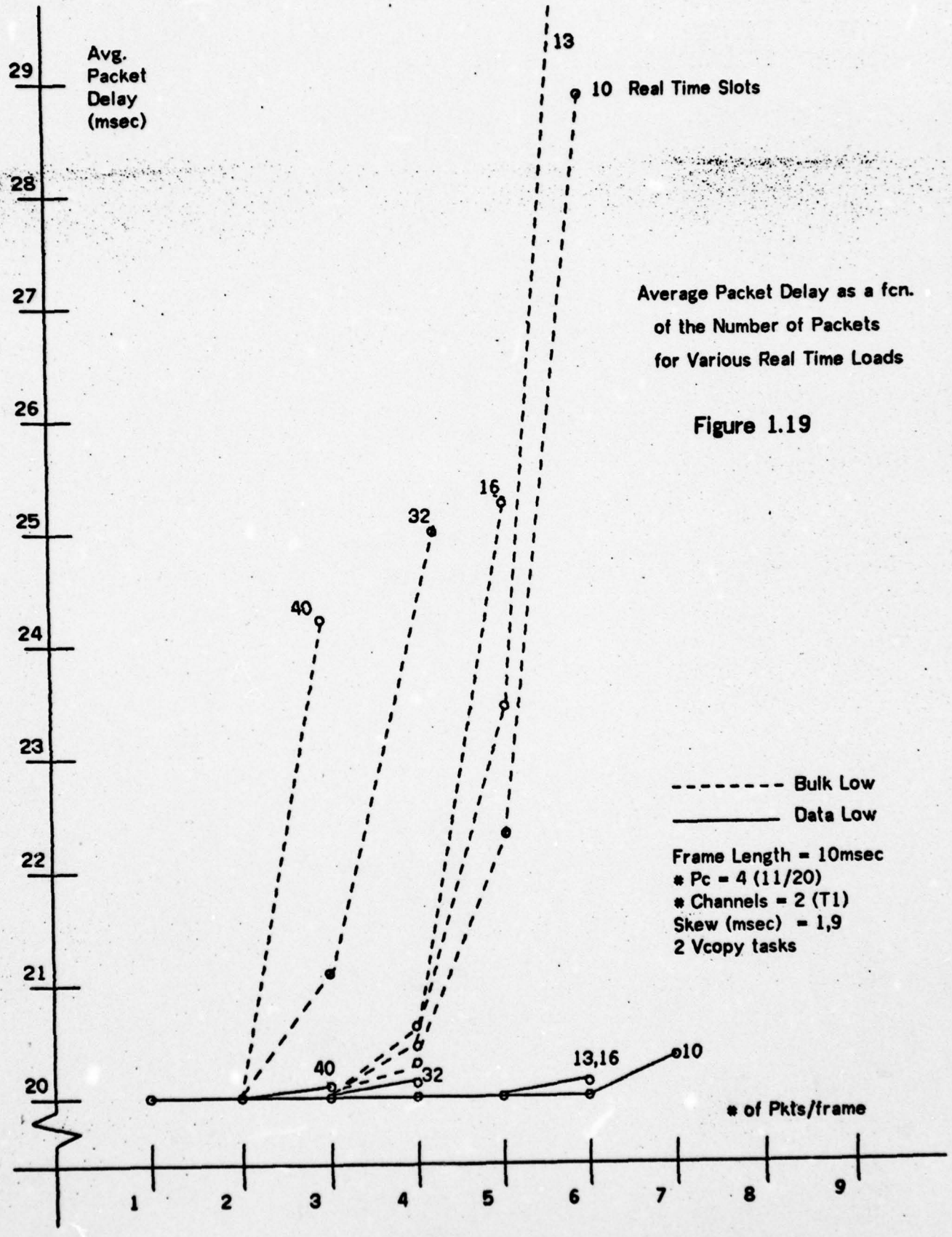
Figure 1.17



Average Packet Delay as a fcn.
of the Channel Skews

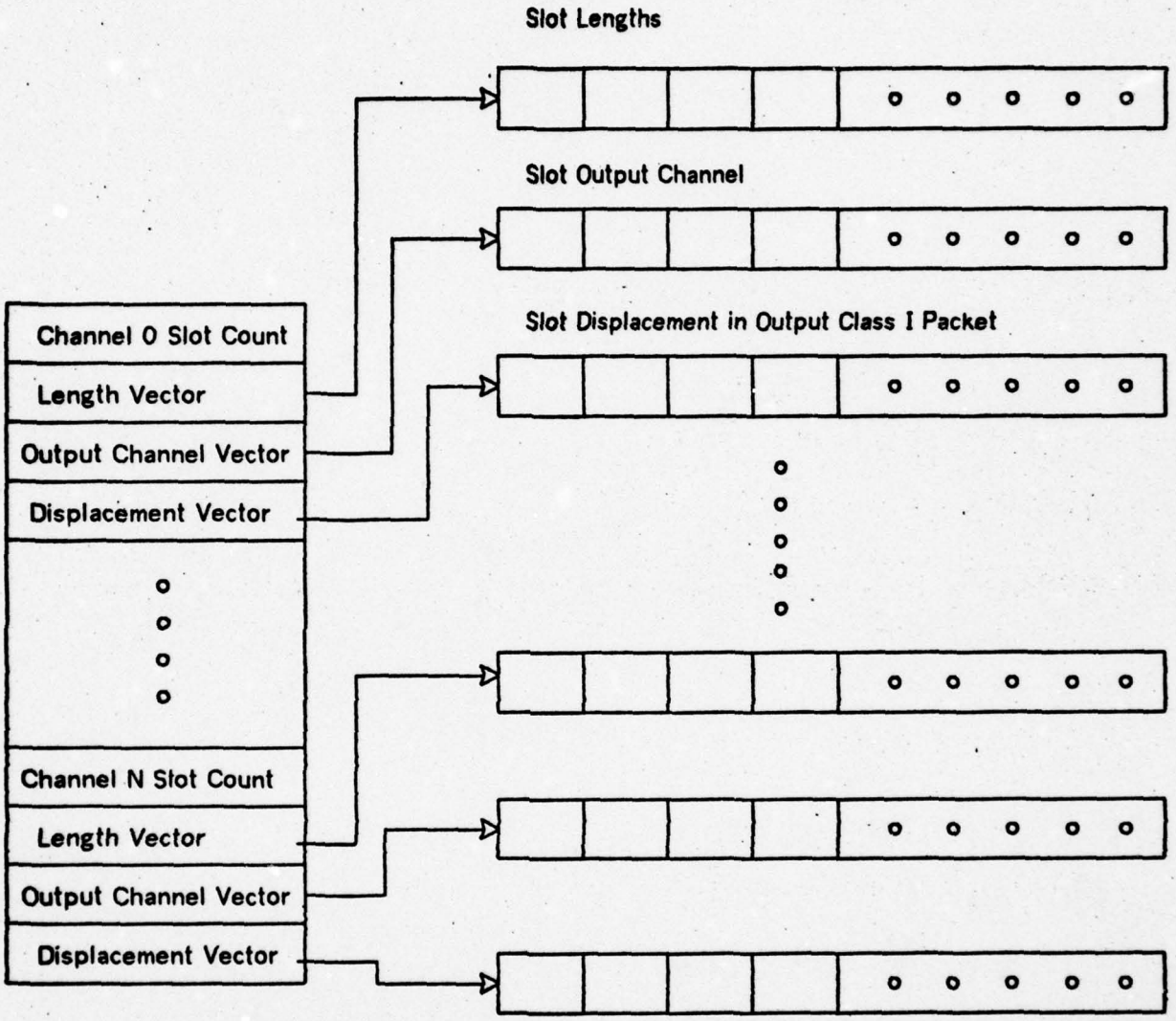
* of Pkts/frame = 1

Figure 1.18



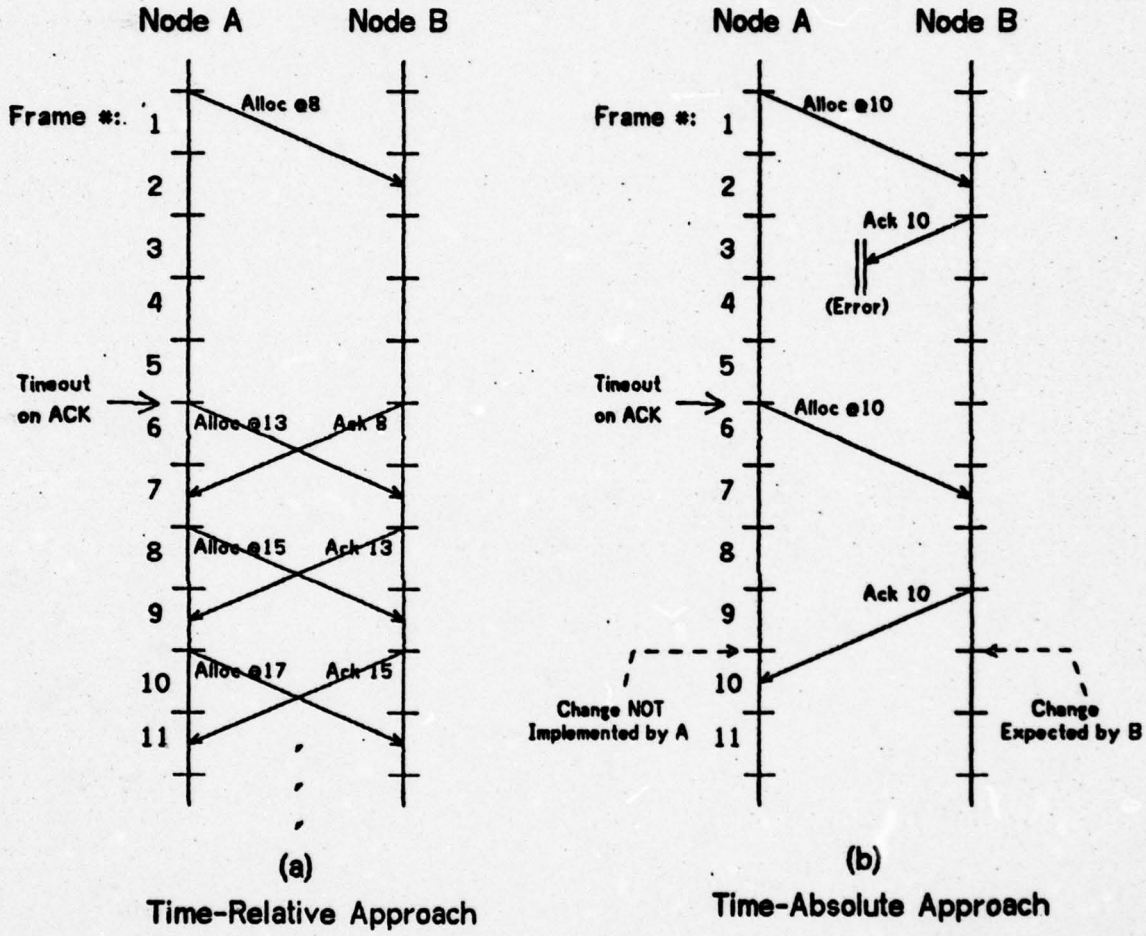
Average Packet Delay as a fcn. of the Number of Packets for Various Real Time Loads

Figure 1.19



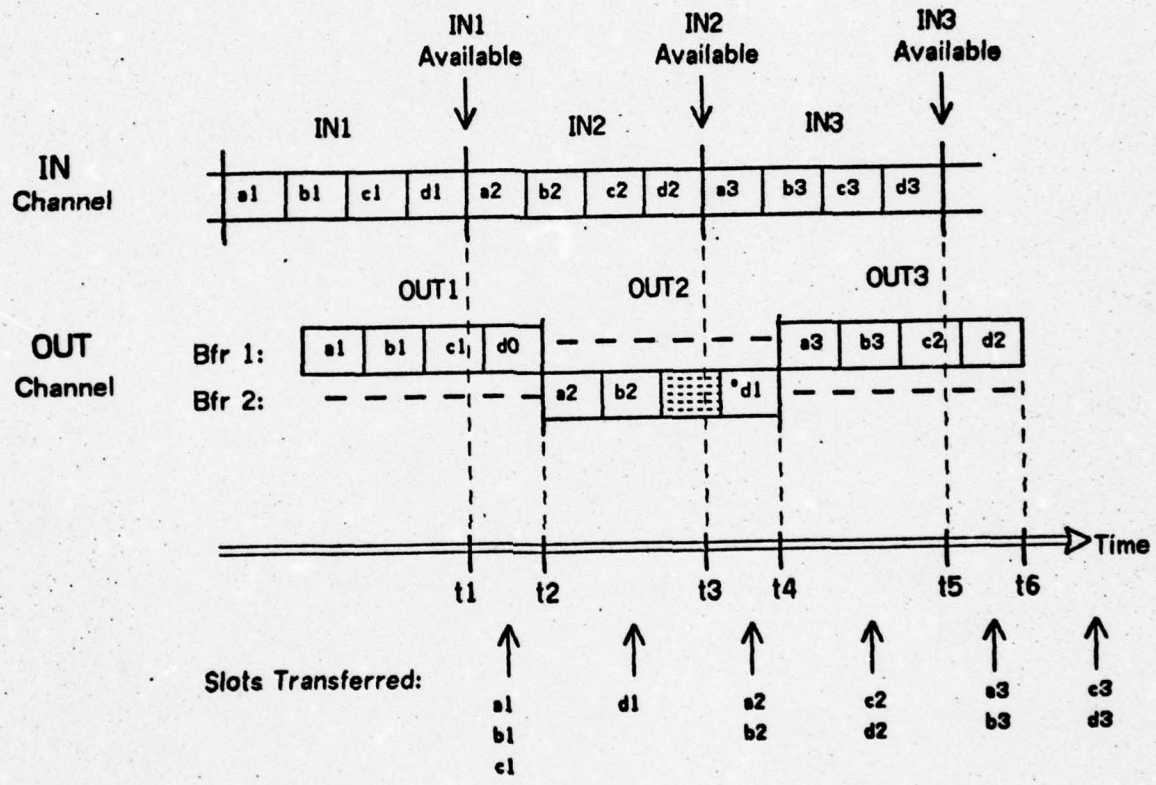
Class I Reservation Tables

Figure A.1



Class I Protocol Error Situations

Figure A.2



Example of Class I Channel Slip

Figure A.3

```
&GO
DO YOU WANT TO MAKE NEW ENTRIES ?  Y

RUN NUMBER  =   245

STARTING SEEDS  =

RUN DURATION TO OBTAIN STEADY STATE IN NUMBER OF FRAMES <3000> = 500

RUN DURATION (AFTER STEADY STATE) IN NUMBER OF FRAMES <3000> =

FRAME DURATION IN MILLISEC. <10>   =

NUMBER OF CHANNELS <4>  =

PLEASE TELL CHARACTERSTICS OF CHANNEL 0
ARE THEY SAME AS OF SOME OTHER CHANNEL  ?   N
SPEED (IN KBDS.) OF CHANNEL 0 <T1>    =

DISTRIBUTION OF PACKET TYPES FOR CHANNEL  0
  ERCENT OF DATA HIGH PACKETS  =  25
PERCENT OF DATA LOW PACKETS   =  25
PERCENT OF BULK HIGH PACKETS  =  25
PERCENT OF BULK LOW PACKETS   =  25

AVERAGE NUMBER OF PACKETS PER FRAME ON CHANNEL 0 <10>    =

MAXIMUM LENGTH IN BYTES OF A PACKET ON CHANNEL 0 <282>    =

MINIMUM LENGTH IN BYTES OF A PACKET ON CHANNEL 0 <26>    =

PERCENT OF PACKETS THAT GO FROM CHANNEL  0
  TO CHANNEL  1   =   100
  TO CHANNEL  2   =
  TO CHANNEL  3   =   0
```

Script Generator Initial Dialog

Figure A.4

```
PLEASE TELL CHARACTERSTICS OF CHANNEL 1
ARE THEY SAME AS OF SOME OTHER CHANNEL  ?
SPEED (IN KBDS.) OF CHANNEL 1 <T1>     =

DISTRIBUTION OF PACKET TYPES FOR CHANNEL  1
  RCENT OF DATA HIGH PACKETS  =  25
PERCENT OF DATA LOW PACKETS   =  25
PERCENT OF BULK HIGH PACKETS  =  25
PERCENT OF BULK LOW PACKETS   =  25

AVERAGE NUMBER OF PACKETS PER FRAME ON CHANNEL 1 <10>     =

MAXIMUM LENGTH IN BYTES OF A PACKET ON CHANNEL 1 <282>     =

MINIMUM LENGTH IN BYTES OF A PACKET ON CHANNEL 1 <26>     =

PERCENT OF PACKETS THAT GO FROM CHANNEL  1
  TO CHANNEL  0   =
  TO CHANNEL  2   =
  TO CHANNEL  3   =   100


PLEASE TELL CHARACTERSTICS OF CHANNEL 2
ARE THEY SAME AS OF SOME OTHER CHANNEL  ?   Y
GIVE ITS NUMBER :  0
AH! I KNOW ABOUT THAT ONE
PERCENT OF PACKETS THAT GO FROM CHANNEL  2
  TO CHANNEL  0   =
  TO CHANNEL  1   =
  TO CHANNEL  3   =   100


PLEASE TELL CHARACTERSTICS OF CHANNEL 3
ARE THEY SAME AS OF SOME OTHER CHANNEL  ?   Y
GIVE ITS NUMBER :  0
AH! I KNOW ABOUT THAT ONE
PERCENT OF PACKETS THAT GO FROM CHANNEL  3
  TO CHANNEL  0   =
  TO CHANNEL  1   =
  TO CHANNEL  2   =   100


PLEASE INPUT COMPOSITION OF REAL TIME SLOT IN RESPONSE TO +
(HIT <CR> TO EXIT)
 NUM FROM TO BYTES<10>
+ 5 0 3
+ 1 3 2 20
+ 6 3 0 10
+ 3 4 2 30
? IMPROPER CHANNEL NUMBER
+ 3 1 3
+
```

Script Generator Initial Dialog

Figure A.5

```
FRAMES DEPARTURE TIME IN MICROSEC. <8000> =
FRAME ARRIVAL TIME IN MICROSEC. ON CHANNEL 0 <0>     =
FRAME ARRIVAL TIME IN MICROSEC. ON CHANNEL 1 <0>     =   3000
FRAME ARRIVAL TIME IN MICROSEC. ON CHANNEL 2 <0>     =   9000
FRAME ARRIVAL TIME IN MICROSEC. ON CHANNEL 3 <0>     =   5000
TIME FOR STATISTICS GATHERING IN MICROSEC. <9000> =
   YOU WISH ME TO DISPLAY YOUR ENTRIES ON TTY ?  Y

R    245          S  16293       0
B   -500          H   3000          F   10          C    4

N0   SPECIFICATIONS FOR CHANNEL  0
T0   1544           A0   10          L0    282    26
P0     25   25   25   25           D0    0  100    0    0

N1   SPECIFICATIONS FOR CHANNEL  1
T1   1544           A1   10          L1    282    26
P1     25   25   25   25           D1    0    0    0  100

N2   SPECIFICATIONS FOR CHANNEL  2
T2   1544           A2   10          L2    282    26
P2     25   25   25   25           D2    0    0    0  100

N3   SPECIFICATIONS FOR CHANNEL  3
T3   1544           A3   10          L3    282    26
P3     25   25   25   25           D3    0    0  100    0

V   COMPOSITION OF REAL TIME SLOT
 NUM FROM TO BYTES<10>
  5    0   3   10
  1    3   2   20
  6    3   0   10
  3    1   3   10

I       0       0       1    3000       3    5000       4    8000       2    9
      5    9000

DO YOU WISH ME TO LIST YOUR ENTRIES ? N
DO YOU WANT TO MAKE ANY CHANGES ?   Y
WHICH ENTRY DO YOU WANT TO CHANGE :  T3
SPEED (IN KBDS.) OF CHANNEL 3 <T1>     =   250

WHICH ENTRY DO YOU WANT TO CHANGE :
DO YOU WISH ME TO DISPLAY YOUR ENTRIES ON TTY ?  N
DO YOU WISH ME TO LIST YOUR ENTRIES ? N
DO YOU WANT TO MAKE ANY CHANGES ?   N
READY TO START...
PAUSE -1 AT: MASTER+170
&
C.MMP LINK RESET, CONNECTION CLOSED
```

Script Generator Initial Dialog

Figure A.6

Chapter 2

Performance Analysis

TABLE OF CONTENTS

CHAPTER 1 PAGE

2.1. Introduction

Several variables are involved in a computer communication switching system, and each one of them may affect the overall performance. A generic configuration of such systems consists of communication lines, transmission control unit, one or more processors, core memory and some auxiliary storage devices such as tapes, drums, and disks. Since the number of parameters is large and each can vary over a vast range, the design and evaluation of such systems is often regarded more as an art than a science. By carefully selecting the simplifying assumptions, a mathematical model can be constructed to describe the influence of each parameter. In this chapter we shall concentrate on three parameters; number of processors, processing speed, and required storage. The most commonly used criterion for performance is average delay through the system. The other criterion used is the probability that delay greater than some fixed value.

For a communication system with FCFS (First-Come-First-Served) discipline, if the distribution of message lengths is the same over all input lines, an integrated system is always better than a segregated one with same total input rate and service rate. Where a segregrated system contains two or more separated input streams and servers with each independent of others. But for different distributions of message length, the integrated system may be worse. These properties have been discussed in [Ver 74]. In the system discussed here, voice and data are of quite different nature. For a 10 millisecond frame and a 8 kbps encoded voice, a call of 3 minutes implies transmission of 18000*80 bits. As the system emulates circuit switching for voice, this is equivalent to a single message of 1260000 bits. Since the maximum data packet

length, is only 2000 bits, the distributions of input messages are substantially different. Evaluation of this kind of voice/data integrated system was first done by Kummerle [Kum 74], then by Fischer [Fis 75]. Both concentrated on the problem of the average delay seen by the data packet.

An outline of the performance model follows. We will use Kendall's notation for queueing model. For the first two parameters "M" refers to the Markovian character of Poissonian arrivals and exponential service, "D" refers the deterministic arrivals and constant service, "GI" refers to the general independent distributions of inter-arrival time and service. The third parameter refers to the number of servers in the queueing system.

General Performance

The integrated switch will be inter-connected in a communication network, termed the backbone system. Local access lines will tie to processors in the backbone. In the backbone system, every Class I or Class II message is assumed to reside in core memory at all times in order to handle the large amount of traffic flow and to satisfy the short delay requirement. Several models have been constructed in an attempt to understand the stochastic behavior of a data packet while it is being processed in the switch.

(i) M/M/y model. There are c processors in the system. The Class I message will be line switched and be assigned a virtual logical channel. A processor can handle p voice channels. Every Class I logical channel consumes 1/p fraction of processing power of a processor. The processing power which is free to handles Class II packets is now y=(c-v/p), where v is the number of logical channels occupied by Class I

messages. This kind of model has been first studied by Kummerle using an approximation algorithm[Kum 74]. Then Fischer and Harris did an exact analysis assuming that y, a random variable, was independent from one frame to the next frame. Both assumed that p=1. In the present case, Class I messages have a long holding time in order to take advantage of line-switching. So y is highly correlated from one frame to another. Later Bhat[Bha 75] did an analysis of multi-channel queueing system with heterogeneous classes, where the number of simultaneous equations to be solved is s(s+1)/2. The exact analysis is given in the next section.

In our experiment, constant Class I traffic is assumed. In the real situation, integrated switch in local loop shall take the responsibility of flow control to keep the backbone system from getting more traffic than their service ability of Class II packets. Therefore, in the following models, the loading factor, which is the ratio of service request of Class II traffic over the corresponding service ability, is assumed to be constant.

Performance model of the experiment

(ii) D+M/M/c model. In this model, the variation of the system loading during a frame period will be discussed. Since Class I traffic is assumed to be constant, b tasks are generated every d seconds. The homogeneous Poisson input of Class II messages generates one task. The service time required by each task of either class has the same exponential distribution. The system will appear different to the tasks of the two different streams. The D stream tasks will see a less crowded system than the Poissonian tasks because of the regularity of the input process of the D stream. The results will show that the difference is significant at low traffic intensities and

diminishes as the traffic intensity increases. The influence of the existence of the D stream is given by an explicit formula. The result can also be used to correct the simulation results when the statistics of the simulation is gathered at some specific point in the frame period.

Internal Structure of Integrated Switch

The above two models discuss the total number of data packets being processed or waiting to be processed in the system. They also dictate the number of buffers needed. However, no information about the structure of the internal task queues was available. The following models try to attack this problem.

Basically there are two types of integrated switch internal structures, distributed , and centralized . In the distributed system, each processor is assigned to some specific task, and each service center has fixed processing power to process its task. While in the centralized system, all processing power is concentrated to service the task of highest priority. Both systems will be discussed in the models that follow.

(iii) Open Network Model. An open system is defined to be one in which the input process is independent of the output process. It is a good model for a processing-bounded system, where the memory is assumed to be very large but the processing power is limited. In this model, the structure and the waiting time of queues are analyzed. The distributed system is modeled as an open queueing network which can be solved by techniques described in [Bas 75] for exponential service request. The centralized system is modeled as an FB_N queueing system which was solved by Schrage[Sch67]

(iv) Closed Network Model. In a closed network, the number of customers in a

system is constant. This is a good model for storage-bounded system, where an incoming message may not find a buffer and has to be neglected. The message which is neglected by the node is not lost in the overall communication. Since the sender cannot receive an acknowledgement of this message, it will send the message again later. But the message does suffer an extra delay because of the finite buffer capacity. If this situation happens often, the delay of message across the network will depend on the period of retransmission of a neglected message. The closed network models are constructed by adding an extra queue, which pools all the empty buffers, into the corresponding open network models. For non-exponential service request of task queues, the approximation techniques described in a series of paper by Chandy et al. [Chan 75A, Chan 75B, Herz 75] are used.

2.2. M/M/y Queueing Model

2.2.1. Introduction

Because of the different the characteristic of data and voice, the integrated switch can be modeled as a queueing system withq heterogeneous classes of customers. So far, systems of this type have not been studied as extensively as those of homogeneous customers. When service rates of two classes are the same, there is no need to distinguish between customer types once a customer joins the system. However, when the service rates of two class are different, the number of the states of the system increases tremendously. This is fully illustrated in Kotiah and Slater [Kot 73] where a two server system of this type has been analyzed. In the integrated switch, different classes of messages not only have different service rates but also have different service disciplines. Voice or the Class I message does not wait, while data or Class II message can be buffered. The system, therefore, becomes very complicated. Kummerle[Kum 74] first gave an approximation algorithm to estimate the average number of packets buffered in the system. Then Fischer and Harris [Fis 75] did an exact analysis assuming that the size of Class I traffic is independent from one frame to another. Later Bhat and Fischer [Bhat 75] used another approach assuming that two classes of traffic compete with each other for s channels. They also gave an approximation algorithm, since for s=20 the number of simultaneous equations is 210. The approximation algorithm approaches the true value when the ratio of Class II message service rate to Class I message service rate is less than 1. When this ratio increases the approximation may differ by one or two orders of magnitude. In the real system we will expect this ratio to be of the order of hundreds.

There are c processors in the system. Each Class I logical channel will subtract 1/p portion of processing power of a processor. There is a maximum number n of logical channels which can be occupied by the Class I jobs. When a Class I job enters the system, if the number of existing logical channels is n, the job will be lost without being serviced. On the other hand, if the existing logical channels is less than n then the job will create a logical channel and will be serviced. The request process of Class I logical channel is a Poisson process with input rate λ_1. The holding time of the logical channel of Class I is exponentially distributed with mean $1/\mu_1$. Class II messages will not create logical channels, but come in by packets. The input process of packets is assumed to be Poissonian with input rate λ_2. A packet can not be serviced by more than one processor simultaneously. The service time of a packet is assumed to be exponentially distributed with mean $1/\mu_2$ by one processor. So if a packet can only get a fractional power of a processor, say 1/d, then the service time of this packet will be d times longer.

2.2.2. Algorithm

The queueing system is a Markov process with state (i,j). Where i is the number of Class I logical channel and j is the number of packets in the system, in service or waiting. The steady state probability of being state (i,j) is $P_{i,j}$. The balance equations for M/M/y model is:

$$(\lambda_1+\lambda_2)P_{0,0} = \mu_1 P_{1,0}+\mu_2 P_{0,1}$$

$$(\lambda_1+\lambda_2+\min(j,d_0)\mu_2)P_{0,j} = \mu_1 P_{1,j}+\lambda_2 P_{0,j-1}+\min(j+1,d_0)\mu_2 P_{0,j+1} \qquad i=0,j>0$$

$$(n\mu_1+\lambda_2+\min(j,d_n)\mu_2)P_{n,j} = \lambda_1 P_{n-1,j}+\lambda_2 P_{n,j-1}+\min(j+1,d_n)\mu_2 P_{n,j+1} \qquad i=n,j>0$$

$$(n\mu_1+\lambda_2)P_{n,0} = \lambda_1 P_{n-1,0}+\min(1,d_n)\mu_2 P_{n,1}$$

$$(\lambda_1+i\mu_1+\lambda_2)P_{i,0} = \lambda_1 P_{i-1,0}+(i+1)\mu_2 P_{i+1,0}+\min(1,d_i)\mu_2 P_{i,1} \qquad 0<i<n,j=0$$

$$(\lambda_1+i\mu_1+\lambda_2+\min(j,d_i)\mu_2)P_{i,j} = \lambda_1 P_{i-1,0}+(i+1)\mu_1 P_{i+1,j}+\lambda_2 P_{i,j-1}+\min(j+1,d_i)\mu_2 P_{i,j+1} \quad 0<i<n, j>0 \qquad 2.1$$

The left-hand side of the above equations are the rate of probability coming into the state (i,j), and the right-hand sides are the rate of probability going out of the state. In the steady state the rates should be balance. Where $d_i=c-i/p$ is the processing power left for Class II packets given that i Class I logical channel exists in the system.

Because the Class I traffic has inherently pre-emptive priority, so for fixed n the GOS (Grade Of Service), which is the probability that a Class I job is not serviced because no logical channel available, is not a function of Class II traffic intensity. GOS is equal to Erlang B loss formula[Sysk 60], i.e.,

$$GOS(n)=(\lambda_1/\mu_1)^n/n!/[\sum_{j=0}^{n}(\lambda_1/\mu_1)^j/j!] \qquad 2.2$$

$$\text{and } P_i=\sum_{j=0}^{\infty} P_{ij}=(\lambda_1/\mu_1)^i/i!/[\sum_{k=0}^{n}(\lambda_1/\mu_1)^k/k!] \qquad 2.3$$

where P_i is the probability that i logical channels exist in the system. Figure 1 and figure 2 shows the design decision of n for certain given GOS value and the function of GOS with respect to traffic intensity and number of channel n.

Define:

$\pi_i(z)=\sum_{j=0}^{\infty} z^j P_{i,j}/\sum_{j=0}^{\infty} P_{i,j}$, generating function of number of packets given that i logical channels exists in the system. Then the set of equation 2.1 can be re-write as follows:

$$[\lambda_1+i\mu_1+\lambda_2(1-z)+d_i\mu_2(1-1/z)]\pi_i(z)-i\mu_1\pi_{i-1}(z)-\lambda_1\pi_{i+1}(z)$$
$$=\sum_{0\le j<(c-i/p)}(d_i-j)\mu_2 z^j(1-1/z)P_{i,j}/\sum_{j=0}^{\infty}P_{i,j} \qquad 2.4$$

define $b_i(z)$ equal to the right hand side of the above equations, and

$$u_i(z)=\lambda_1+i\mu_1+\lambda_2(1-z)+d_i\mu_2(1-1/z) \qquad 0\le i<n$$
$$u_n(z)=n\mu_1+\lambda_2(1-z)+d_n\mu_2(1-1/z) \qquad 2.5$$

above equations can be re-written in the matrix form as follows:

$$A(z)\pi(z)=(1-1/z)b(z) \qquad 2.6$$

where

$$\pi(z)=[\pi_0(z)\ \pi_1(z)\ \ldots\ldots\pi_n(z)]^T$$

$$b(z)=[b_0(z)\ b_1(z)\ \ldots\ldots b_n(z)]^T$$

and

$$A(z)=\begin{pmatrix} u_0(z) & -\lambda_1 & & & & \\ -\mu_1 & u_1(z) & -\lambda_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -i\mu_1 & u_i(z) & -\lambda_1 & \\ & & & \ddots & \ddots & -\lambda_1 \\ & & & & -n\mu_1 & u_n(z) \end{pmatrix}$$

The above equation can be solved by conditions that

(i) $z=1$, $\pi_i(1)=1$

(ii) $\pi_i(z)$ is analytical in the region $0\le z<1$

Define matrix R(z) with (i,j) elements $r_{i,j}(z)$

$r_{i,j}(z)=(-1)^{i+j}$(cofactor of $a_{j,i}(z)$, the (j,i) element of matrix A(z))

Then R(z)A(z)=A(z)R(z)=determinant of A(z)=|A(z)|

$$\pi(z)=\frac{R(z)}{|A(z)|}(1-1/z)b(z) \qquad 2.7$$

Theorem 1: |A(z)|=determinant of A(z) has exactly n different roots in (0,1), and z=1 is also a root.

This theorem can be proved similar to the methods of [Mitr 68]. The basic

argument follows: $|A(1)|=0$ is trivial for the row summation of A is zero. Define $g_i(z)$ as the principal minor of order i of A(z). $|A(z)|$ is a polynomial of degree (2n+2); using standard techniques, such as counting the number of sign changes in the sequence $g_i(z)$, i=0,1,2,..,n at z=0 and z=1, one can determine the number of zeros of $|A(z)|$ in (0,1). If $z_j^{(i)}$ is the jth root of $g_i(z)$, then for a special property of A(z) that consecutive $g_{j+1}(z_j^{(i)})$ will have difference signs. So $z_j^{(i+1)}$, roots of $g_{i+1}(z)$, will be in $(z_{j-1}^{(i)}, z_j^{(i)})$ interval for j=0,1,..,i. With conditions that $z_0=0$, only one root of $g_0(z)$ is in (0,1), and 1 is the root of $g_n(z)=|A(z)|$, the proof can be carried by the above interval partition procedure. The same procedure is also done numerically. *

Theorem 2: $R(z_i)$ has rank one for z_i, a root of $|A(z)|$ in (0,1].

The proof is very tedious but straightforward. Basically the proof is calculating the following formula

$$r_{jk}(z_i)r_{ml}(z_i)-r_{jl}(z_i)r_{mk}(z_i)$$

Each element in the above formula is a minor of A(z). By expanding the determinants, term by term they will combine with each other to a form of $p(z_i)\cdot|A(Z_i)|$, so the formula can be proved equal to zero for any j,k,m,l. So the matrix R is of rank 1 for z, a root of A(z). *

Because R(z) has rank 1, so we can choose arbitrary row of R(z) as r(z). By the above theorems, we have (n+1) equations:

$$r(z_i)b(z_i)=0 \qquad 2.8$$

for i=1,2,..,n

and

$$r(1)b(1)=\frac{d}{dz}A(z)|_{z=1} \qquad 2.9$$

The unknown variables are all $P_{j|i}$ for $0\le j<d_i$, n_3 are defined as the number of these variables. Where $P_{j|i}=P_{i,j}/\sum_{j=0}^{\infty} P_{i,j}$. While number of equations in 2.4 for all $0\le j<d_i$ is $n_3-(n+1)$. So together with the above (n+1) equations, $P_{j|i}$ can be solved. The mean number of packets given that there are i logical channels, $\pi_i'(1)$, can be solved by following method.

As equations 2.4 and 2.6, $A(z)\pi(z)=(1-1/z)b(z)$

Take derivative with respect to z of the above equation

$$A'(z)\pi(z)+A(z)\pi'(z)=1/z^2\cdot b(z)+(1-1/z)b'(z) \qquad 2.10$$

then

$$\pi'(z)=\frac{R(z)}{|A(z)|}[1/z^2\cdot b(z)+(1-1/z)b'(z)-A'(z)\pi(z)] \qquad 2.11$$

Take limit as z approach to 1,

$$\pi'(1)=1/[\frac{d}{dz}|A(z)|_{z=1}]\cdot\{R'(z)[b(z)/z^2-A'(z)\pi(z)]$$

$$+R(z)\cdot[-2b(z)/z^3+2b'(z)/z^2-A''(z)\pi(z)-A'(z)\pi'(z)]\}_{z=1} \qquad 2.12$$

Then

$$\pi'(1)=\{R'(1)[b(1)-A'(1)\pi(1)]+R(1)[-2b(1)+2b'(1)-A''(1)\pi(1)-A'(1)\pi'(1)]\}/a_1$$

where $a_1=\frac{d}{dz}|A(z)|_{z=1}=\sum_{j=0}^{n} r_{nj}(-\lambda_2+d_j\mu_2)$,

$A'(1)$ and $A''(1)$ are diagonal matrix with ith element $(-\lambda_2+d_i\mu_2)$ and $(-2d_i\mu_2)$ respectively. Then $\pi_i'(1)$ can be written in the following form:

$$\pi_i'(1)=1/a_1\cdot\{\phi_i+a_2+\sum_{j=0}^{n} g_j\pi_j'(1)\} \qquad 2.13$$

where $\phi_i=\sum_{j=0}^{n} r_{ij}'(1)[b_j(1)+\lambda_2-d_j\mu_2]$

$$a_1=\sum_{j=0}^{n} r_{nj}(1)[-\lambda_2+d_j\mu_2]$$

$$a_2=\sum_{j=0}^{n} r_{nj}(1)[-2b_j(1)+2b_j'(1)+2d_j\mu_2]$$

All the above parameters are known, $\pi_i'(1)$ the mean number of packets for i logical channel can then be calculated.

2.2.3. Numerical Consideration

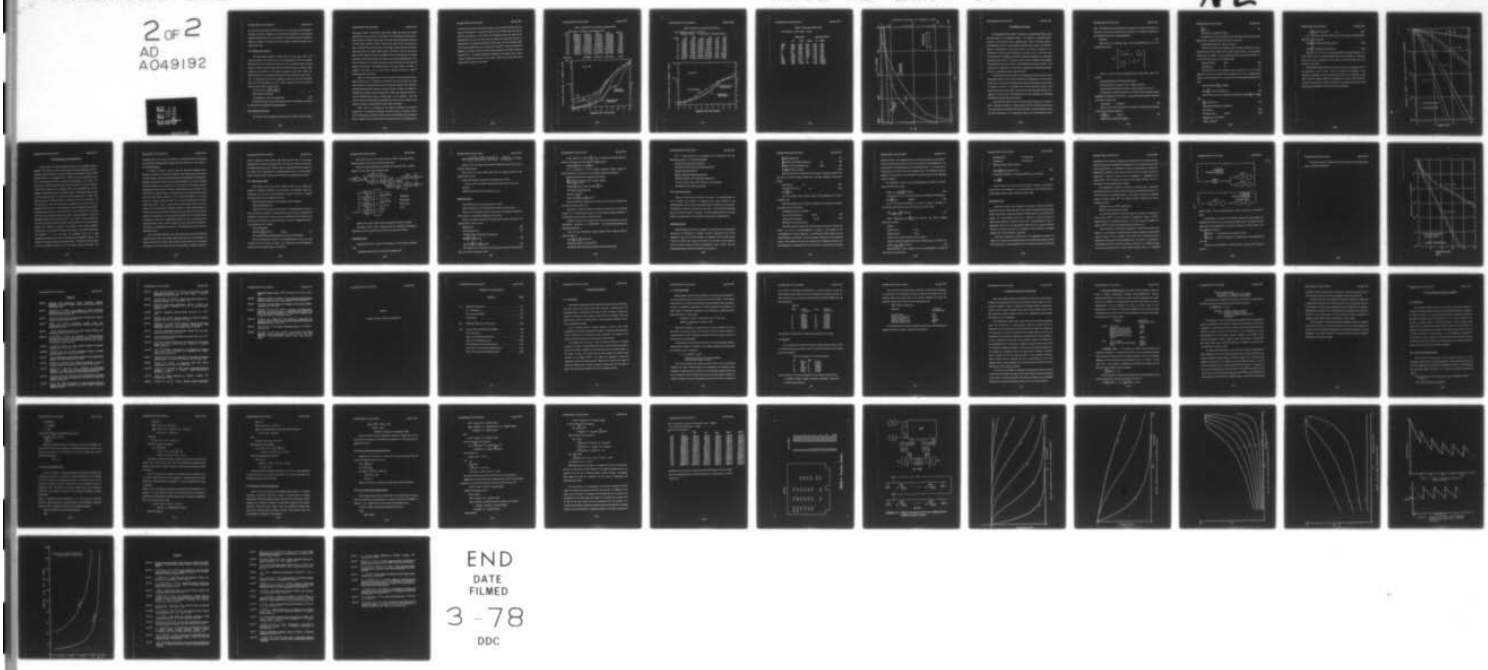
Number of simultaneous equations has to be solved by the above algorithm depends on p and c. For p=1, the special case, the system is similar to system analyzed in [Bhat 75]. The number of simultaneous equations is (n+1)n/2. But for c=1, n_3 equal to n, the maximum number of logical channels.

The algorithm discussed in the last section will be numerically stable for μ_1 and μ_2 in the same order of magnitude. As the ratio μ_1/μ_2 increases, the algorithm becomes less stable. Double precision, or even quadruple precision if available, is suggested for μ_1/μ_2 in the order of thousands. Calculating the roots of |A(z)| in (0,1) and calculating $r(z_i)$ are the two most sensitive steps. Principally, |A(z)| is a polynomial of z of order 2n+2. The coefficients of this polynomial can be calculated, and the polynomial solved using a standard package. This method is not recommended for large μ_1/μ_2. The reason is that all the variables of |A(z)| are in the form of a large number plus or minus a small number. Multiplication and division of such numbers are numerically unstable. For example (1.001*1.002*1.003) can not be distinguished from $(1.002)^3$ for a machine with 20 bits for mantissa. More accurate but quite costly is substituting the iterative z into |A(z)| directly and calculating the determinant. Calculation of $r(z_i)$, principally, is calculating the principal minor, which is the determinant of a principal submatrix of $A(z_i)$. The argument of instability is the same. Because $R(z_i)$ has rank 1, there are a lot of redundancy in $R(z_i)$. The right choice of a

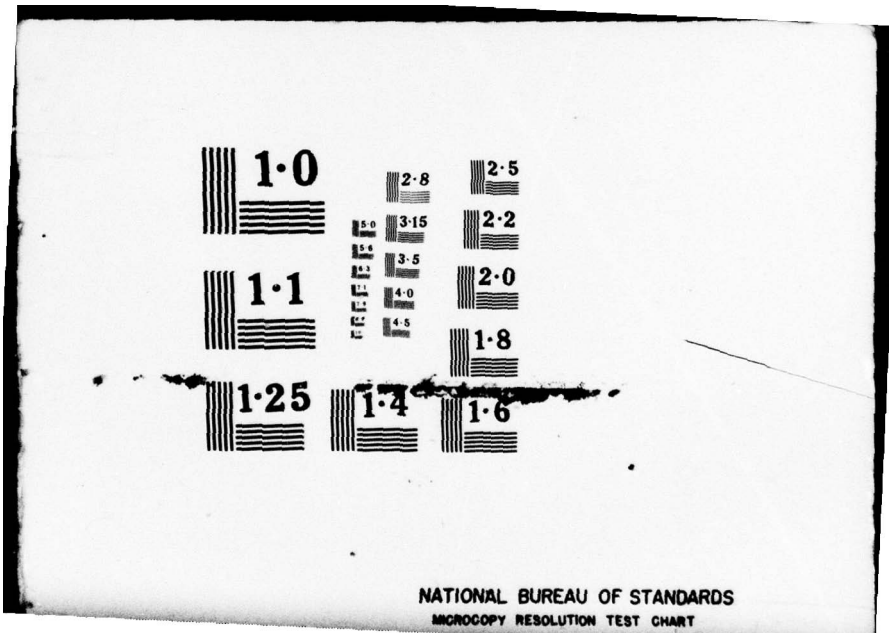
AD-A049 192 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 17/2
THE APPLICATION OF MULTIPLE PROCESSOR COMPUTER SYSTEMS TO DIGIT--ETC(U)
JUN 76 M BARBACCI, L CHANG, S FULLER, A INGLE DCA100-75-C-0066

UNCLASSIFIED SBIE-AD-E100 014 NL

2 OF 2
AD
A049192



END
DATE
FILMED
3-78
DDC


1·0
1·1
1·25
2·8
3·15
3·5
4·0
4·5
5·0
2·5
2·2
2·0
1·8
1·4
1·6
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

row will yield a much better result than others. As a rule of thumb, if the computation of $|A(z_i)|$ is backward iteration from n to 0, then the computation of $r(z_i)$ should be backward iteration from 0 to n, and vice versa. The backward iteration algorithm is used for its stability. The details of the error analysis in backward iteration can be found in [Jone 74].

2.2.4. Approximation Algorithm

The approximation algorithm of [Bhat 75] fail for large μ_1/μ_2, which is the practical situation. We will give here a more complicated but much more accurate approximation for the special case where c=1. From the numerical result of above analysis, it shows that the average number of packets for μ_1/μ_2 and heavily loaded system are not sensitive to c, the number of processor in the system. Where the heavily loaded system is defined that the input rate of class II traffic is greater than c-n/p. It is equivalent to say that for segregated systems with same total processing power with the same degree of service for Class I system, then the Class II system will have traffic intensity greater than one.

c=1, $d_i=1-i/p$, and $g_i=\sum_{j=0}^{\infty} jP_{i,j}/\sum_{j=0}^{\infty} P_{i,j}$

Then equation 2.4 becomes

$$-(\lambda_1+i\mu_1)g_i+i\mu_1 a_{i-1}+\lambda_1 g_{i+1}=-\lambda_2+d_i\mu_2(1-P_{0|i}) \qquad 2.14$$

Assuming that $P_{0|i}=1/(1+g_i)$ and substituting to the above equations, we will get g_i. We call this the conditional mean approximation.

2.2.5. Comparison of Results

For constant λ_1/μ_1 and λ_2/μ_2, the average number of packets increases rapidly

when μ_2/μ_1 increases. This behavior is also shown in [Bhat 75]. There is no channel reserved only for data packets in that article. In our model, when the input rate of Class II packets is greater than d_n, which is the worst case processing power for Class II traffic, similar behavior is shown. Otherwise, the system is not very sensitive to μ_2/μ_1. Notice that when $d_i > \lambda_2$ for some i, the average number of packets will be quite large when there are i Class I logical channel. The reason is that once the system goes into this state, the input rate is greater than the processing rate, so packets accumulate. Until sometimes later, the system can give more processing power to the Class II packets for the releasing of some Class I logical channels, then the number of packets will decrease. The integrated switch has a large variation of number of packets in the system. This is in some sense a trade off between the usage of processing power and memory.

Table 1 is the average number of Class II packets in system given that i Class I logical channels exist. For different c, the number of servers, $\pi_i'(1)$ differs very little from each other on overload states, which are states $d_i \leq \lambda_2$. Because of this character, although the approximation algorithm is based on uni-server system, mean number for overload states should be a good approximation. Another very important character is that the conditional mean number of packets varies a lot for different i. Even with rather small mean waiting length, the probability of overflow, which is defined as the number of packets in system exceeds its buffer space, is quite high.

Table 2 and 3 are comparison to the results of Kummerle's and Fischer's respectively. In Kummerle's approximation, the service scheme is the same as ours, but the service time of packets is constant rather than exponential. Both Kummerle's and our results show the sharp increase of the conditional mean in overloaded states.

In table 2 we also note that the mean increases as μ_2/μ_1 increases although λ_1/μ_1 and λ_2/μ_2 are kept constant. In Fischer and Harris's TN 6-75 DCEC report, they fail to show this characteristics. In their later TC 21-75 DCEC report, the new model has this characteristics, but they only tend to give the total mean waiting time while as we stated before is not enough to describe the whole system. The TC 21-75 DCEC report is the special condition where p=1 and cp=n of our analysis. The numerical results of [Bhat75] , which does not allow the Class II packet preemptive the Class I traffic, are compared here. The blocking probability of [Bhat75] is much higher than our result because in [Bhat75] no priority is given to Class I traffic, for the same reason it has much shorter waiting line in the queue.

Table 1 Comparison of exact solution and approximation

cp=15; n=10; λ_1/μ_1=5.0; $\lambda_2/c\mu_2$=7/15; μ_1/μ_2=100;

π_i(1)	P_i	c=1	c=3	c=5	Approx.
i= 0	.0068315	0.8881841	1.5899381	2.4360016	.876
i= 1	.0341575	1.0399755	1.7091119	2.5267855	1.002
i= 2	.0853938	1.2894188	1.9175224	2.6971034	1.170
i= 3	.1423231	1.7669488	2.3417091	3.0699985	1.405
i= 4	.1779038	2.7981681	3.3011327	4.0220692	1.762
i= 5	.1779038	5.1520923	5.5606589	6.2629728	2.399
i= 6	.1482532	10.4236670	10.7955900	11.3819070	4.188
i= 7	.1058951	21.0693230	21.4009170	21.9430840	11.637
i= 8	.0661845	38.8442910	39.1415930	39.6471080	29.404
i= 9	.0367691	61.7339510	62.0130820	62.4995080	53.226
i=10	.0183846	81.8106690	82.0841770	82.5715340	73.894
Total mean		11.93888501	12.38936477	13.05113306	8.195

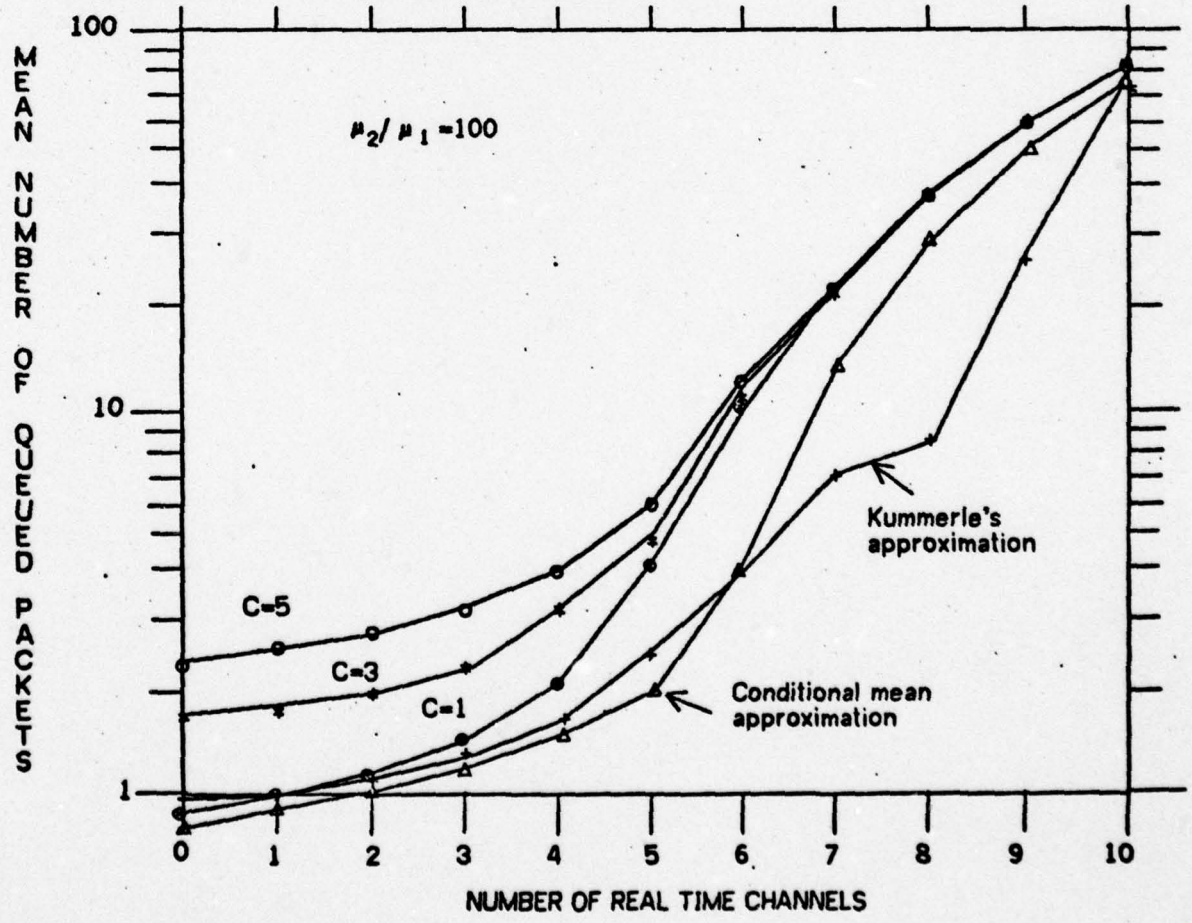


Table 2 Comparsion of Kummerle's approximation

for 15 channels; $\lambda_1/\mu_1=5.0$; $\lambda_2/\mu_2=7/15$;

	Kummerle's Approx.		Exact Calculation		Cond. Mean Approx.	
μ_2/μ_1	10	100	10	100	10	100
0	.9375	.9375	.92832	.88818	.891	.876
1	1.0714	1.0714	1.10053	1.03998	1.022	1.002
2	1.2500	1.2500	1.36056	1.28942	1.202	1.170
3	1.5000	1.5000	1.76762	1.76695	1.469	1.405
4	1.8750	1.8750	2.40928	2.79817	1.908	1.762
5	2.5000	2.5000	3.39682	5.15209	2.696	2.399
6	3.7500	3.7500	4.84210	10.42367	4.072	4.188
7	7.5000	7.5000	6.81077	21.06932	6.175	11.637
8	8.4999	8.4999	9.25161	38.84429	8.888	29.404
9	11.7024	27.7738	11.89144	61.73395	11.814	53.226
10	18.0428	73.3285	14.03198	81.81067	14.144	73.894

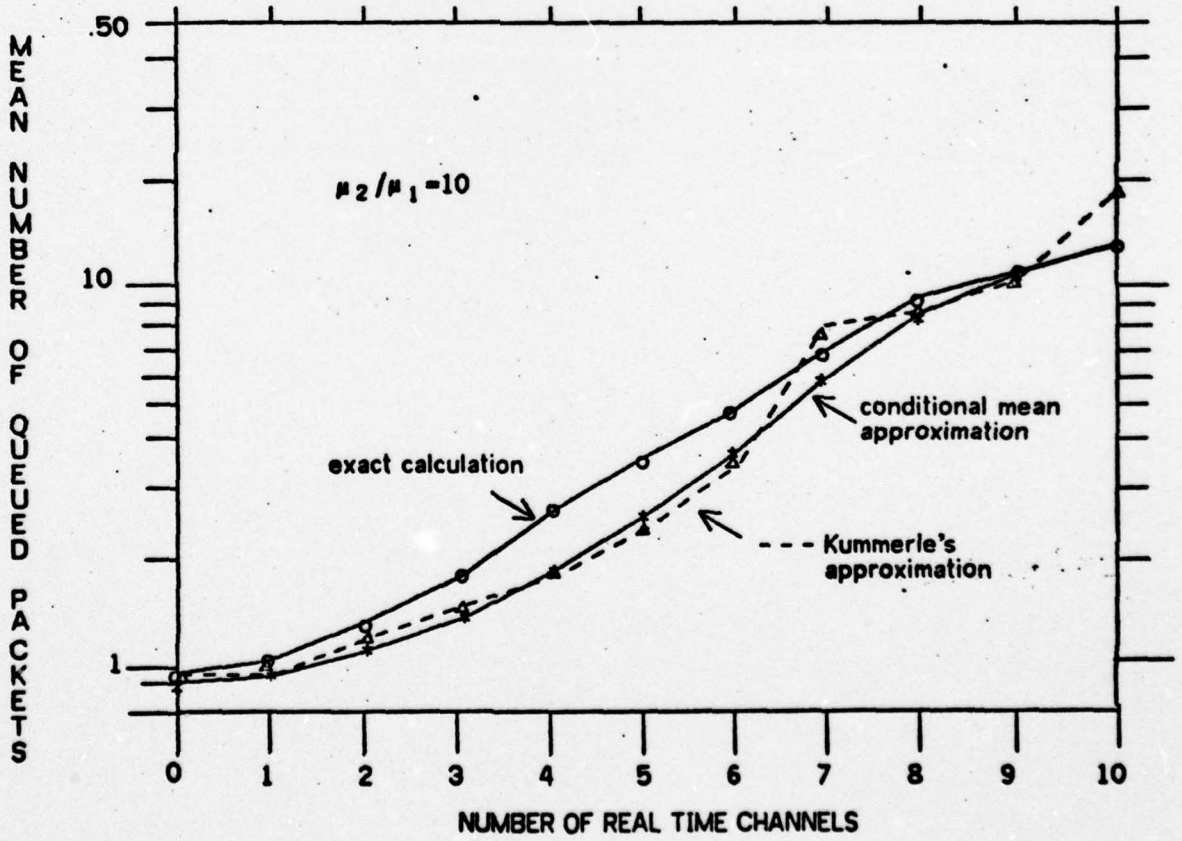
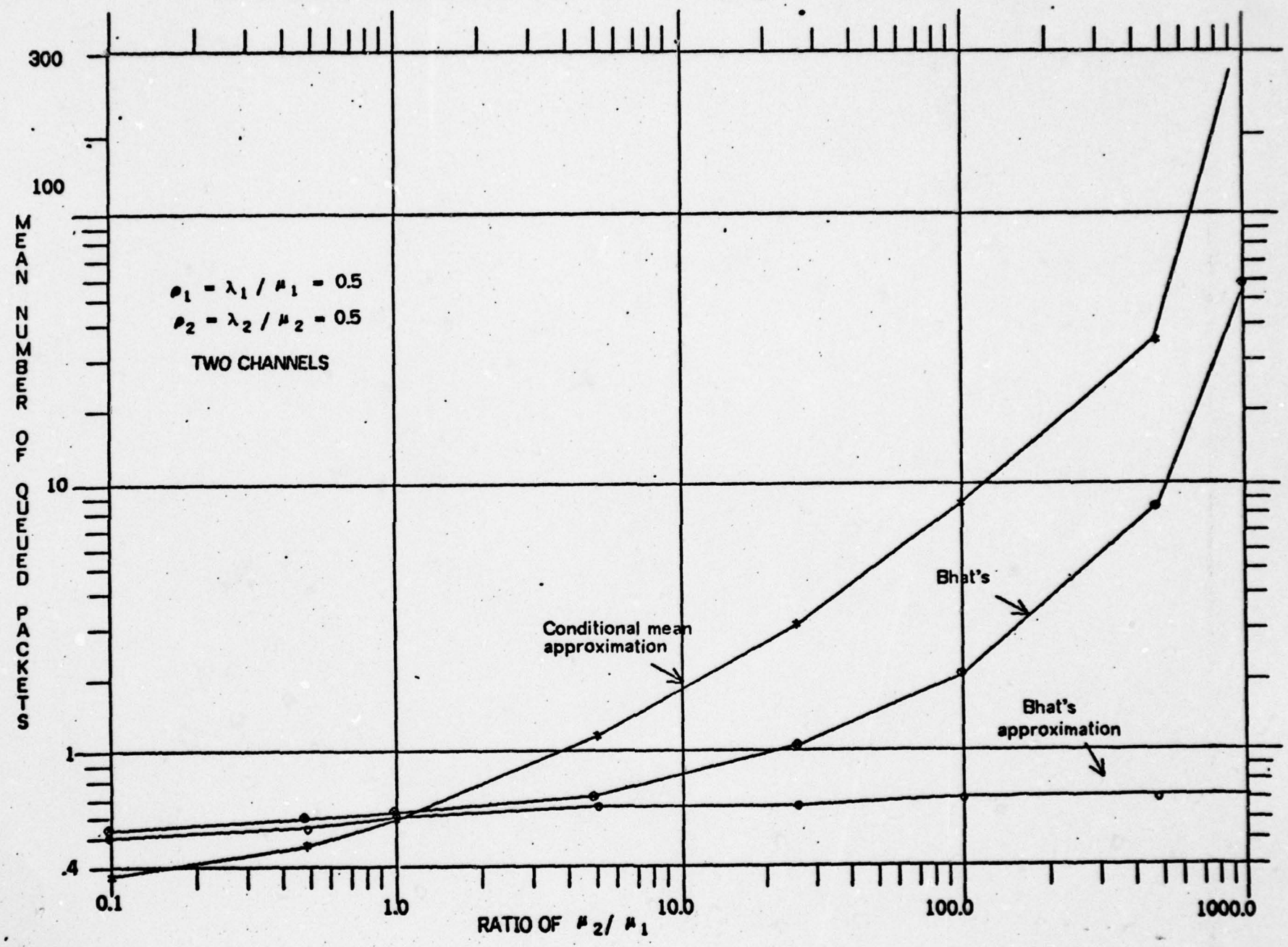


Table 3 Comparsion of Bhat's model

for 2 channels, ρ_1=0.5, ρ_2=0.5, $\alpha=\mu_2/\mu_1$

	Bhat's Model			Cond. Mean Approx.	
α	PB	$E[Q_2]$	$E_a[Q_2]$	PB	$E_a[Q_2]$
.0005	.2461	.5333	.5001	.1429	.3335
.005	.2461	.5336	.5005	.1429	.3352
.05	.2462	.5368	.504	.1429	.3517
.5	.248	.5618	.5505	.1429	.4867
1	.25	.5833	.5833	.1429	.6026
5	.2574	.6945	.6605	.1429	1.205
25	.2629	1.046	.6965	.1429	3.071
100	.2646	2.212	.705	.1429	8.585
500	.2651	8.34	.71	.1429	35.835
1000	.2652	15.994	.71	.1429	69.552
5000	.2652	77.219	.705	.1429	338.839


300
100
10
1
.4
MEAN NUMBER OF QUEUED PACKETS
$\rho_1 = \lambda_1 / \mu_1 = 0.5$
$\rho_2 = \lambda_2 / \mu_2 = 0.5$
TWO CHANNELS
Conditional mean approximation
Bhat's
Bhat's approximation
0.1
1.0
10.0
100.0
1000.0
RATIO OF μ_2 / μ_1

2.3. D+M/M/c Queueing Model

The integrated switching system is modeled as a queueing system whose inputs are composed of two independent streams. The concept of a piecewise Markov process is used to analyze such a queueing system. One of the streams (which is corresponding to the flow of data packets) is Poissonian with intensity λ, and the other (which is corresponding to the constant level of voice slots) is assumed to be a deterministic renewal process with intensity ν and batch of size b. The service times of all the tasks are independent and identically distributed according to an exponential distribution with mean μ^{-1}. This kind of system has been analyzed by Kuczura[KUCZ 72] as GI+M/M/c case with c=1 and ∞. We will generalize the result to arbitrary c and a batch input of magnitude b. The GI stream is assumed to be a deterministic process, for every d second there will be a batch of b tasks.

The state of the system (number of tasks waiting and in service) seen by a task will generally differ from that of the D stream tasks. Consequently, in a system where tasks wait for service such as D+M/M/c queue, the service received by the two types of customer will differ. For non-batch and first-come-first-served system, the D stream tasks always receive better service than the Poissonian tasks because of its regularity of input process. While significant at low traffic intensities, this advantage diminishes as the traffic intensity increases.

Let Y(t) be the number of task in the system (those waiting and in service) at time t. Since the D stream is a renewal process and Y(t) is Markovian between any two consecutive arrival epochs of the tasks, $\{Y(t), t\geq 0\}$ is a piecewise Markov process with state space$\{0,1,2,\ldots\}$. The degeneration points are the arrival epochs of the D

tasks. The Markov process operating with the segments is a birth-death process with birth rate λ and death rate μ, the same for all segments. The transition probability can be calculated numerically by following algorithm:

$$p'(t)=Ap(t) \qquad 3.1$$

where p(t) is the probability column vector $p_i(t)=Prob[Y(t)=i]$ and A is the infinitesimal generator.

$$A^T = \begin{pmatrix} -\lambda_1 & \lambda_1 & & & \\ \mu_1 & -\mu_1-\lambda_1 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_i & -\mu_i-\lambda_1 & \lambda_1 \\ & & & \ddots & \ddots \end{pmatrix}$$

where $\mu_i = \min(i, c)*\mu$ is the processing rate of system when i jobs in the system.

$$p(t)=\exp(At)p(0)=[I+(At)+(At)^2/2!+(At)^3/3!+\ldots]p(0)$$

P(t) is the transition probability matrix = transpose of exp(At).

$P_{i,j}(t)$=transition probability from state i to state j during a time interval t.

Let $\{p_j\}$ be the stationary distribution of the Markov chain imbedded at points immediately preceding a D task arrivals. If $r_{i,j}$ is the one-step transition probability from state i to state j, then

$$r_{i,j}=P_{i+b,j}(d) \qquad i,j=0,1,2,3,\ldots \qquad 3.2$$

where d is the inter-arrival time of the D tasks. The distribution $\{p_j\}$ satisfies the Chapman-Kolmogorov equations.

$$p_j=\sum_{i=0}^{\infty} p_i r_{ij}, \qquad j=0,1,2,\ldots \qquad 3.3$$

subject to the normalization condition

$$\sum_{j=0}^{\infty} p_j = 1 \qquad 3.4$$

is the stationary distribution of $\{p_j\}$.

Let $\{q_j\}$ be the stationary distribution of the state seen by an arbitrary, arriving Poisson tasks. Using rate conservative principle we can write:

$$\nu p_i + \lambda q_i + \min(i, c)\mu q_i = \min(i+1, c)\mu q_{i+1} + \lambda q_{i-1} + \nu p_{i-b}$$

where the left-hand is the asymptotic rate of transition leaving state i and the right-hand side is the asymptotic rate of transition coming to state i. No simple closed form solution is available. It can be calculated numerically without much difficulty. For b=1, the closed form solution is available as follow:

$$\nu p_j + \lambda q_j = (j+1)\mu q_{j+1} \qquad 0 \le j < c \qquad 3.5$$

$$\nu p_j + \lambda q_j = c\mu q_{j+1} \qquad j \ge c \qquad 3.6$$

The left-hand side is the asymptotic rate of transition from j to j+1 and the right-hand side is the asymptotic rate of transition from j+1 to j. Multiplying both sides by u^j and summing over all j, we obtain the following relation between p(u) and q(u).

$$\nu p(u) + \lambda q(u) = c\mu[q(u) - \sum_{i=0}^{c-1} (1-i/c)q_i u^i]/u \qquad 3.7$$

$$q(u) = [\sum_{i=0}^{c-1} (1-i/c)q_i u^i + u\sigma p(u)]/(1-\rho u) \qquad 3.8$$

where $\sigma = \nu/c\mu$ and $\rho = \lambda/c\mu$. Applying the normalization condition $q(1)=1$. We will get:

$$\sum_{i=0}^{c-1} (1-i/c)q_i = 1-\rho-\sigma \qquad 3.9$$

along with the following (c-1) equations.

$$\mu q_1 - \lambda q_0 = \nu p_0 \qquad 3.10$$

$$(i+1)\mu q_{i+1} - \lambda q_i = \nu p_i \qquad 0 \le i \le c-2 \qquad 3.11$$

$q_0, q_1, q_2 \ldots q_{c-1}$ can be solved.

$$q_i = [\nu p_{i-1} + \lambda q_{i-1}]/c\mu$$

$$=\sigma\sum_{j=c-1}^{i-1} p_j\rho^{i-j-1}+q_{c-1}\rho^{i-c+1} \qquad i\geq c \qquad 3.12$$

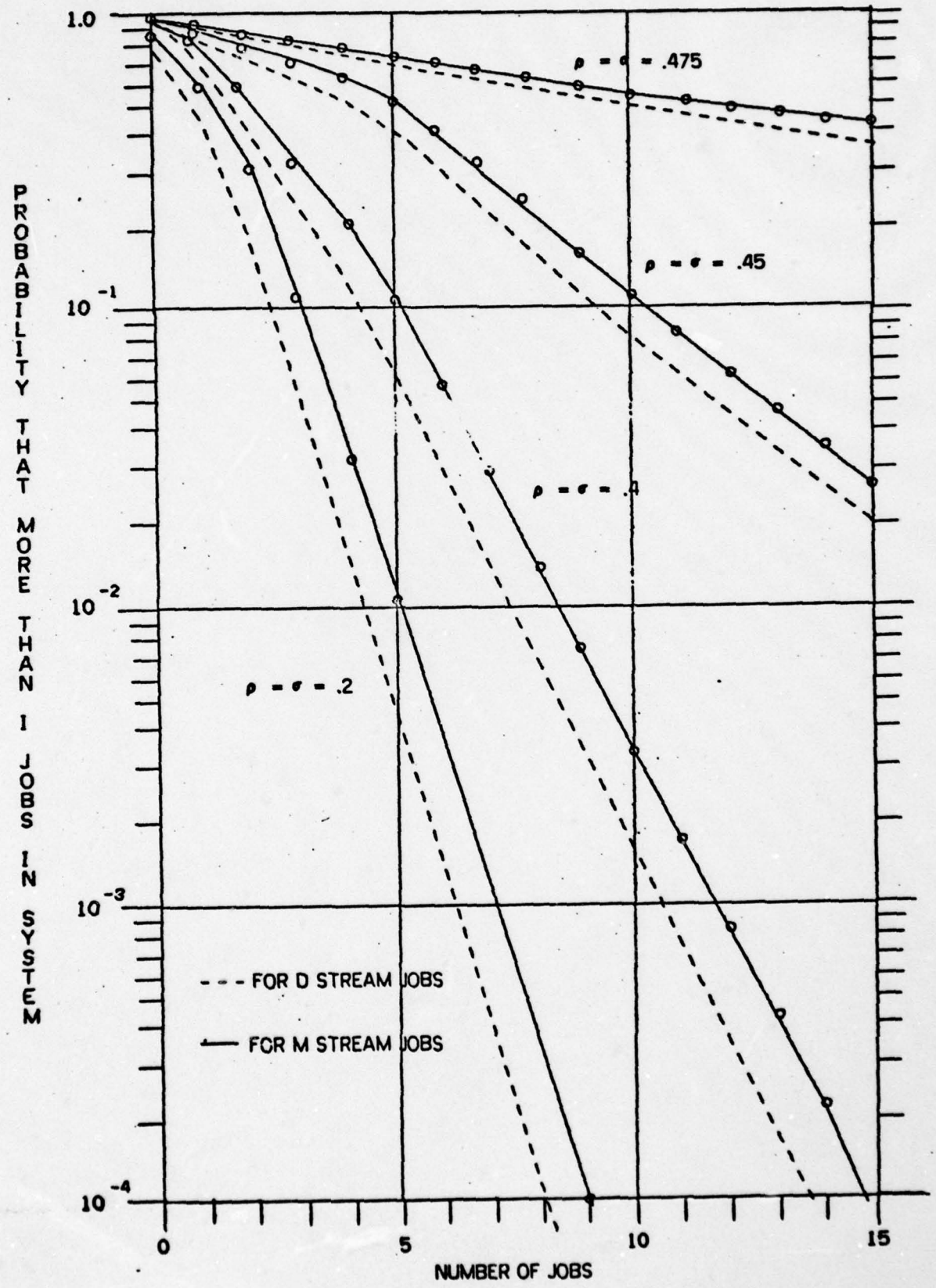
If l_1 is the mean of the distribution $\{p_j\}$ and l_2 is the mean of the distribution $\{q_j\}$, then $l_1=p'(1)$, and $l_2=q'(1)$.

$$q'(1)=[\sum_{i=0}^{c-1}(1-i/c)iq_i+\sigma+\sigma p'(1)]/(1-\rho)+\rho/(1-\rho) \qquad 3.13$$

$$l_2=[\rho+\sum_{i=1}^{c-1}(1-i/c)iq_i]/(1-\rho)+\sigma(1+l_1)/(1-\rho) \qquad 3.14$$

Note that the first term on the right-hand side of the above equation is the mean number of tasks in the system M/M/c queue with traffic intensity ρ. Hence, the second term may be considered to be an increase in the mean number due to the presence of the D tasks.

This D+M/M/c model describes the effect on the system by changing the level of the deterministic stream of class 1 tasks in the multi-server environment. If the statistics are gathered at the end of each frame, then $\{p_j\}$ will be calculated. On the other hand, if the statistics gathering are triggered by the input of Class II packets or by some random snap shot, $\{q_j\}$ will be calculated. From another point of view, if both $\{p_j\}$ and $\{q_j\}$ are gathered, μ, the service rate, can be estimated.


1.0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
ρ = σ = .475
ρ = σ = .45
ρ = σ = .4
ρ = σ = .2
- - - FOR D STREAM JOBS
—— FOR M STREAM JOBS
PROBABILITY THAT MORE THAN I JOBS IN SYSTEM
0
5
10
15
NUMBER OF JOBS

2.4. Internal Structure of Integrated Switch

In this section, the internal queueing structure of the integrated switch is discussed. The model and results heavily depends on the task decomposition and the processor-memory structure of the system. Barbacci and Oakly [Barb 76] give a detail task decomposition of an integrated switch. That work is primarily designed for C.mmp. The task decomposition scheme is assumed to be generalized for any system. The other factor is the PMS (Processor Memory Switch) structure of the system. There are various PMS structures for an integrated switch. We will partition them into two categories, distributed system and centralized system. The distributed system implies that the processing power is restricted to do some specific tasks. A typical well-known example of a distributed system is a computer with a front-end processor. The main processor or the front-end processor in generally will not do the tasks pre-assigned to the other even it is idle and capable. A more complicated system is CM* of Carnegie-Mellon University . Each processor has its own local memory in which some but not all of the application programs are stored. Although processors can fetch the application program stored in global memory or local memory of other processors, it will suffer extra delay through the memory mapping mechanism. So in the distributed system, each processor is assumed to be assigned to some specific application programs, and it will only execute tasks corresponding to it. In the centralized system, the processor will execute any tasks. The centralized system has a large homogeneous memory which stores all the buffers and application programs. The ordinary uni-processor system is the typical example. The processor(s) can be located to do any application program at any time without suffering extra delay. The

processing power of the system is centralized. For some instant, all the processor(s) may execute the same application program, while this situation can never happen in the distributed system.

A network of queues is used to model the distributed integrated switch. Networks of queues have been widely studied as a multi-programmed and time-shared computer systems. Jackson [Jack 63] and Gordon and Newell [Gord 67] developed the equilibrium distribution of states of a class of general networks. In particular, Gordon and Newell make clear the product form of the solution of the balance equations describing the steady state of the model. In these models the service center can be connected in any arbitrary fashion. A customer leaving a service center simply choose the next service center according to a set of branching probability for the center being left. Jackson's model also allows for the arrival and departure of customer outside the system. There is a big constraint in these models that all the service request must be exponentially distributed. Baskett et al [Bask 75] generalized the networks containing several class of customers and non-exponential service time for special service discipline. The result also has the product form. In a series of paper of Chandy et al [Cha 75], an approximation algorithm to solve a more general non-exponential service closed network is discussed. The first iteration of the algorithm is the solution for exponential service network. These techniques are used to solve the distributed system.

For a centralized integrated switch, models of FB_N queue system and a $M/M/1/\lambda_i$ queue system are used. Where FB_N queueing system consists of a service center and queues of different service time. When a customer finishes the service in ith queue, it will branch to the (i+1)st queue with some fixed probability. $M/G/1/\lambda_i$ queueing is

used to modeled the closed network model where the input rate, λ_i, of the queue dependents on the state of the queue which is the number of customer in the system. As stated before, open network model is used for analyzing the service structure of the system and is appropriate for processing-bounded system. A closed network model is used for buffer contention and is appropriate for storage-bounded system.

2.4.1. Open Network Model

From theorem 6.3 of [Jack 63] for constant arrival rate of a system, the equilibrium probability distribution of queue length at the individual centers are independent and also each of these distribution is identical with that for a isolated queueing system with some equilibrium input rate.

The routing-generating process is specified by a set of parameters,

$R=\{r(m,n)|m\in[0,M]\wedge n\in[1,M+1]\}$

where M is the total number of service center. The interpretation in that for m and n $\in$[1,M]: (i) the probability is r(0,n) that Center n will be first on a routing, (ii) the probability is r(m,M+1) that if Center m is ith on a routing, then this ith element is the last one, (iii) the probability is r(m,n) that if Center m is ith on a routing, then there is a (i+1)st element and it is Center n.

The set of equations

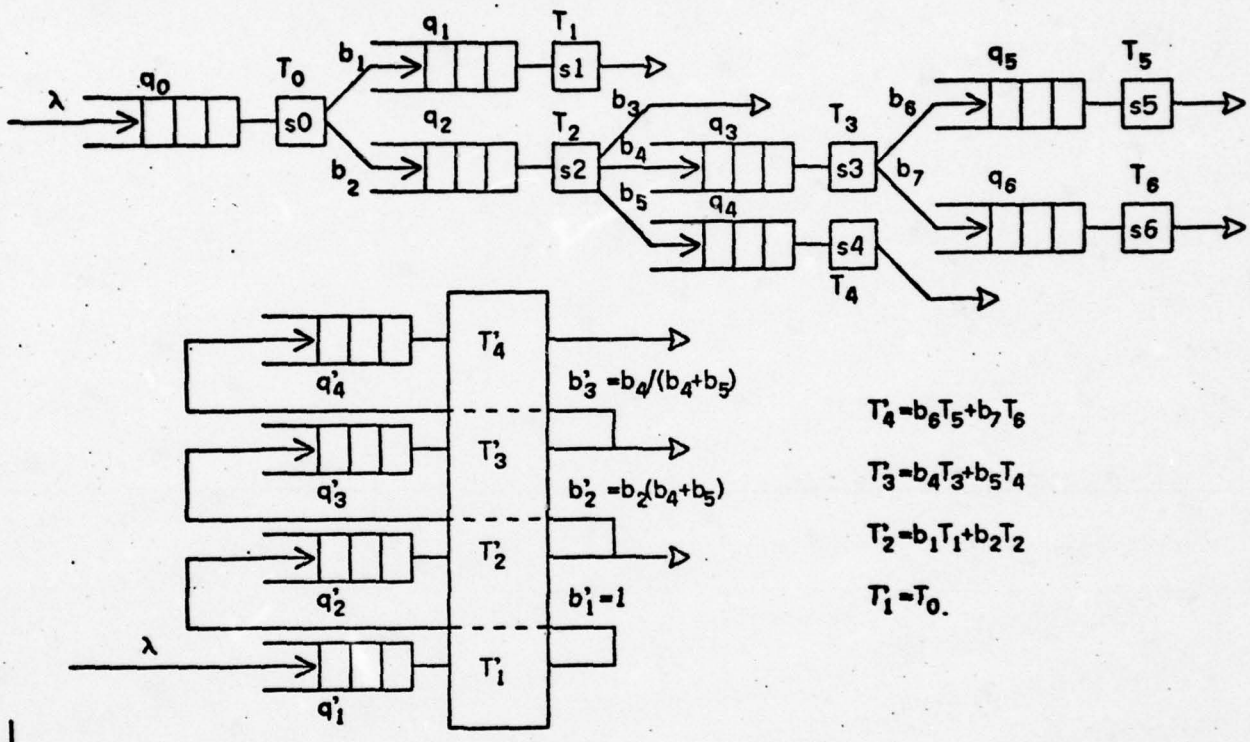
$$e(n)=r(0,n)+\sum_{m=1}^{M} e(m)r(m,n), \qquad n\in[1,M] \qquad 4.1$$

has a unique solution $\{e(n)|n\in[1,M]\}$; and the e(n) are all non-negative.

The task decomposition scheme of the integrated switch, as Fig.1, is a tree structure, e(i) can be solved very easily. The transformation from the distribution system to the centralized system is as in the figure:

T_i=the service time of the ith task queue; the length of processing which a customer request in the ith queue of the distributed system.

T_i'=the ith simple processing time; the length of processing which a customer requests in the ith queue of the centralized system.



Similar formula can be written if the task decomposition scheme is modified.

and $e(0)=1$, $e(1)=b_1$, $e(2)=b_2$, $e(3)=b_2b_4$, $e(4)=b_2b_5$, $e(5)=b_2b_4b_6$, $e(6)=b_2b_4b_7$, because the physical meaning of e(i) are the net input rate of the ith queue.

Centralized System

Define W_n'=the time in system at the completion of the nth simple processing time.

This FB_N queueing system was solved by Schrage[Schr 67].

$$E[W_n'] = \frac{[\lambda E(U_n^2) + \sum_{K=n+1}^{\infty} A_K E(T_K'^2)]}{2[1-\lambda E(U_{n-1})][1-\lambda E(U_n)]} - \frac{E(S_{n-1})}{[1-\lambda E(U_{n-1})]} - E(T_n') \quad 4.2$$

where T_n' = the nth simple processing time; the length of processing which a job in the nth queue receives.

$S_n = T_1' + T_2' + \ldots + T_n'$ service request given that the customer returned to the system at least n times.

$U_n = S_n$ if the customer returned to the system at least n-1 items,

$= T_1' + T_2' + \ldots + T_K'$ if the customer returned to the system only k-1 times, k<n.

$A_k = \lambda \Pi\, b'_j$

Defined $S_0 = U_0 = 0$, and $b'_0 = 1$, then $U_n = U_{n-1} + b'_{n-1} T_n'$

Distributed System

Here exponential service time of each task is assumed.

Define $\rho_i = \lambda e(i)/(\mu_i c_i)$ is the equivalent traffic intensity of service center i.

Where c_i is the processing power in service center i and the service request at service center i is exponentially distributed with mean $1/\mu_i$.

Network R is fixed for some fixed task decomposition scheme, so does e(i). The average length of queue i will be:

$$E[l_i] = \rho_i/(1-\rho_i) \quad 4.3$$

$$\text{and } Var[l_i] = \rho_i/(1-\rho_i)^2 \quad 4.4$$

The total number of customers in the system l,

$$E[l] = \sum_{i=1}^{M} E[l_i] = \sum_{i=1}^{M} \rho_i/(1-\rho_i) \quad 4.5$$

$$\text{and } Var[l] = \sum_{i=1}^{M} Var[l_i] = \sum_{i=1}^{M} \rho_i/(1-\rho_i)^2 \quad 4.6$$

The optimum way to distribute the processing power depends on the criterion used. Here several examples are given:

If the criterion is to minimize $\sum_{i=1}^{n} l_i$, that is to minimize the average number of packets in the system, with the condition that $\sum_{i=1}^{n} C_i$=constant.

minimize $\sum_{i=1}^{n} E[l_i]$; given that $\sum_{i=1}^{n} C_i=C$.

It is a transform of the link capacity assignment problem studied by Kleinrock[Klei 63]. The optimize way is so called the square root assignment.

Let C_i=processing power assigned to service center i.

$\sum_{i=1}^{M} C_i=C$=total processing power of system =constant.

The problem becomes:

Minimize $\sum_{i=1}^{M} \lambda_i/(\mu_i C_i-\lambda_i)$ given constraint $\sum_{I-1}^{M} C_i=C$.

The optimum assigning becomes

$C_i=\lambda_i/\mu_i+\alpha(\lambda_i/\mu_i)^{1/2}$

where $\alpha=(C-\sum_{i=1}^{M}(\lambda_i/\mu_i))/\sum_{i=1}^{M}(\lambda_i/\mu_i)^{1/2}$.

Other criterion, like minimizing the kth moment of l_i, or the minimax criterion can be used as referred to [Meiz 72].

Here gives a numerical example to calculate the waiting time of a centralized system and a distributed system given that the processing capacity assignment is minimizing the average number of packets in system.

Suppose C=5, λ_0=1.0, b_1=.4, b_2=.6, b_3=.4, b_4=.4, b_5=.2, b_6=.5, b_7=.5, T_0...T_7 are exponentially distributed, i.e. $E[T^2]=2\cdot(E[T])^2$. and $E[T_0]=E[T_1]=E[T_4]=E[T_5]=1.0$, $E[T_2]=E[T_3]=E[T_6]=2.0$.

From the task decomposition scheme, e(0)=1.0, e(1)=.4, e(3)=.24, e(4)=.12, e(5)=.12, e(6)=.12,

$\alpha=C-\sum_{i=0}^{6} e(i)T_i/\sum_{i=0}^{6}[e(i)T_i]^{1/2}=0.4$

Waiting $W_i=1/(\mu_i C_i-\lambda_i)=1/(\alpha[\lambda_i/T_i]^{1/2})$

W_0=2.5, W_1=3.95, W_2=4.56, W_3=7.22, W_4=7.22, W_5=7.22 W_6=10.20;

For centralized system, we can calculate the b_i' according to the task decomposition graph. b_1'=1.0, b_2'=.36, b_3'=.6667,

$E[T_1']$=.2, $E[T_2']$=.32, $E[T_3']$=.2, $E[T_4']$=.3,

$E[T_1'^2]$=.08, $E[T_2'^2]$=.1664, $E[T_3'^2]$=.0672, $E[T_4'^2]$=.14,

$E[S_1]$=.2, $E[S_2]$=.52, $E[S_3]$=.72,

$E[U_1]$=.2, $E[U_2]$=.52, $E[U_3]$=.592, $E[U_4]$=.792,

$E[U_1^2]$=.08, $E[U_2^2]$=.3744, $E[U_3^2]$=.458, $E[U_4^2]$=.7573,

From the formula of FB_N system waiting time can be calculated:

W_1=.5042, W_2=.9740, W_3=2.54, W_4=6.525,

2.4.2. Close Network Model

There are a finite number of buffers in the system. If an incoming packet can not be located with a buffer, it is neglected by the switch. Because of the acknowledgement feature of the communication network, this packet will be sent again. This packet is not lost but it does suffer an extra delay. In this section the probability that this situation happens is calculated. The same queueing network of the last section is studied except for an extra queue for pooling the unoccupied buffers.

Centralized System

When the total service time of a customer in the integrated switch is exponential distributed, the probability of i customer in switch is irrelevant to their service discipline as long as the service discipline is independent of the service request. There are n buffers and c servers in the switch. Let the state, i, be the number of buffer currently occupied, and $P_i(t)$ be the probability of state i at time t. Then the backward Kolmogorov differential equations can be written as:

$$\frac{d}{dt}P_0(t)=-\lambda P_0(t)+\mu P_1(t) \qquad 4.7$$

$$\frac{d}{dt}P_i(t)=\lambda P_{i-1}(t)-(\lambda+i\mu)P_i(t)+(i+1)\mu P_{i+1}(t) \qquad i<c \qquad 4.8$$

$$\frac{d}{dt}P_i(t)=\lambda P_{i-1}(t)-(\lambda+c\mu)P_i(t)+c\mu P_{i+1}(t) \qquad n>i\geq c \qquad 4.9$$

$$\text{and } \frac{d}{dt}P_n(t)=\lambda P_{n-1}(t)-c\mu P_n(t) \qquad 4.10$$

Define the steady state probability P_i=limit of $P_i(t)$ as t approach to infinite. The left side of the above equations become zero. The above equations can be solved easily:

$$P_i=\lambda^i/(i!\mu^i)\cdot P_0 \qquad i<c \qquad 4.11$$

$$P_i=\lambda^c/(c!\mu^c)\cdot(\lambda/c\mu)^{i-c}\cdot P_0 \qquad i\geq c \qquad 4.12$$

$$\text{with } \sum_{i=0}^{N} P_i=1,$$

P_n is the probability that a customer comes in and is neglected for the lack of available buffer.

Define S_n=probability that a customer is neglected for a system with n buffers, then $S_n=P_n$(given that n buffers).

The following relations are derived:

$$S_{n+1}=1/[1+(n+1)\mu/(\lambda S_n)] \qquad \text{for } n<ci \qquad 4.13$$

$$\text{and } S_{n+1}=1/[1+c\mu/(\lambda S_n)] \qquad \text{for } n\geq c \qquad 4.14$$

With $S_0=1$ as the initial condition.

The above analysis assumes that once the packet is sent out, it relieves the buffer. It is true if no acknowledgement is required or the response of the acknowledgement is almost immediately. In the following, the finite response time of acknowledgement is considered. Let l be the number of output links of this integrated switch. Those packets which have been processed by this integrated switch are sent out through the corresponding links to the next switch. Let q_0 and S_0 be the queue

and service time of this integrated switch. q_1,..until q_l are the queues corresponding to the links through which packets are sent out and waiting for acknowledgements. The (l+1)st queue is the pooling queue for the unoccupied buffers. The service rate of (l+1)st queue is the input rate of this integrated switch. The probability that zero customer in (l+1)st queue is the probability that a packet coming into the integrated switch is neglected.

For exponential response time of acknowledgement, the Gordon and Newell's result of product form is used:

$$P(n_0,n_1,..,n_{l+1})=\frac{1}{G(N)}\prod_{i=0}^{l+1}[(X_i)^{n_i}/A_i(n_i)] \qquad 4.15$$

where $(X_0,X_1,..,X_{l+1})$ is a real positive solution to the eigenvector-like equations,

$$\mu_j X_j=\sum_{i=0}^{l+1}\mu_i X_i p_{ij} \qquad 0\le j\le(l+1) \qquad 4.16$$

p_{ij} are the branching probability from queue i to queue j and G(N) is a normalizing constant defined so that all $P(n_0,..n_{l+1})$ sum to one. That is,

$$G(N)=\sum_{n\in S(N,l+1)}\prod_{i=0}^{l+1}[(X_i)^{n_i}/A_i(n_i)] \qquad 4.17$$

where $S(N,M)=\{(n_0,n_1,..,n_{l+1})|\sum_{i=0}^{l+1}n_i=N$ and $n_i\ge 0\ \forall i\}$. and $A_i(n)$ is defined recursively as follows:

$A_k(0)=1$,

$A_k(n)=n\cdot A_k(n-1)$ if $n<c_k$

$A_k(n)=c_k\cdot A_k(n-1)$ if $n\ge c_k$

where c_k is the number of server in service center k.

Algorithm derived by Buzen[Buz 73] is used. Assuming $X_0,..X_{l+1}$ are solved. Let

$$g(n,m)=\sum_{n\in S(n,m)}\prod_{i=0}^{m}[(X_i)^{n_i}/A_i(n_i)] \qquad 4.18$$

Note that G(N) is equal to g(N,l+1), and, in fact, g(n,l+1)=G(n) for n=0,1,2,..N. From the above equations that

$g(n,0)=X_0^n/A_0(n)$ for n=0,1,2,..,N.

and $g(0,m)=1$ for m=0,1,..,l+1.

With the recursive relation of g(n,m):

$$g(n,m)=\sum_{k=0}^{n}(X_m)^k/A_m(k)\cdot g(n-k,m-1) \qquad 4.19$$

g(n,m) can be calculated. The marginal probability p_{l+1} is calculated.

$$P_{l+1}=\frac{g(N,l)}{G(N)} \qquad 4.20$$

Figure 3 shows the result of the above algorithm. Compare to the result of system without acknowledgement, it is not surprised that the finite response time of acknowledgement gives a higher demand of the buffer space.

Distributed System

There are N buffers and M queues in the system. There are well-known algorithm to solve the exponential service time. Gordon and Newell [Gord 67] gave the product form solution. Buzen [Buz 73] gave an efficient algorithm to solve this kind of system. For non-exponential service time , Chandy et al [Chan 75] gave an approximation algorithm to solve this kind of system. The approximation is exact for exponential service time. The approximation algorithm uses the concept of Norton's Theorem for electrical circuit theory. The algorithm is fully elaborated in the papers of [Chan 75]. A brief introduction is given here.

The idea of complement of queue is used. The complement of a queue i in a network R is a queue which completely captures the interface between queue i and the rest of the network. For queueing network of exponential service time, local balance

is satisfied and the complement of queue can be determined. But for general network (which do not satisfy local balance), the complement can not be determined exactly and must be approximated. The complement of a queue is approximated by a queue with independent exponential service time; the service rate for this queue, however, is appropriately adjusted to account for the assumptions of independence and exponentiality. This is referred by Chandy as local balance interface, since it is computed by using assumptions of local balance.

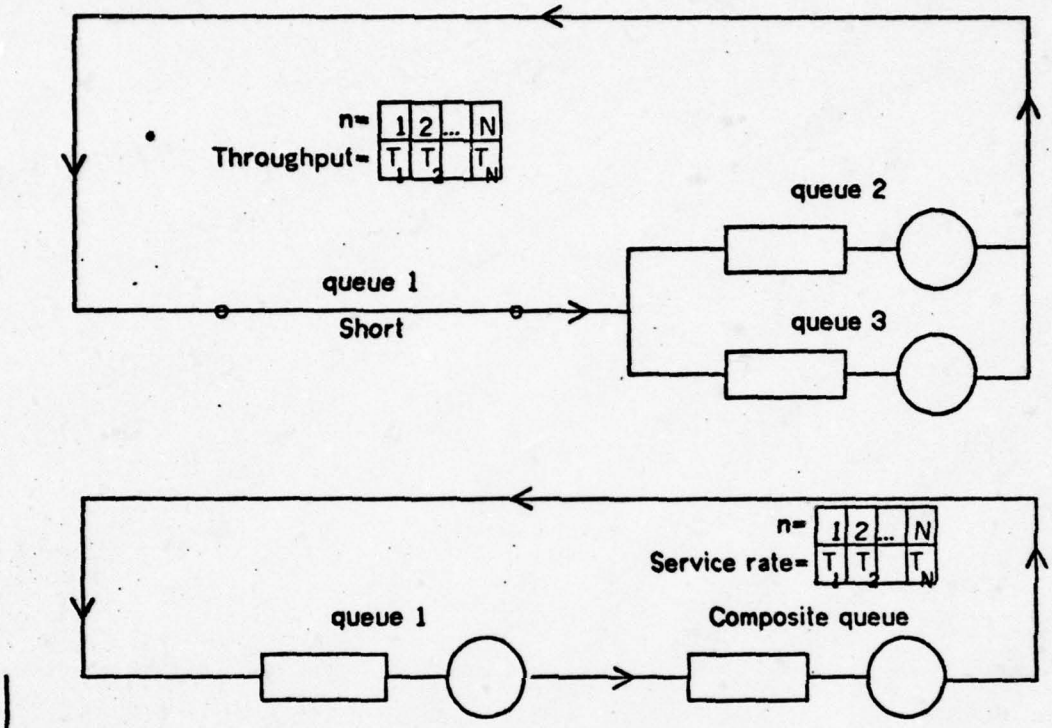
The complementary algorithm is carried out in iterative steps , sequentially adjusting local balance interface to better approximate the complementary queues. On the kth iteration of the algorithm k=0,1,2,... local balance interfaces are calculated using an auxiliary network $S^{(k)}$. The auxiliary network is identical to the actual network except that

(i)the service rates in $S^{(k)}$ are different and

(ii)$S^{(k)}$ is assumed satisfy local balance.

It is therefore possible to use Norton's Theorem to compute the complement of every queue in $S^{(k)}$. At the kth iteration the complement of queue i in $S^{(k)}$ is approximated to the complement of queue i in R. In other words, the complement of queue i in $S^{(k)}$ is used as the local balance interface of queue i in R.

The two networks in the figure are equivalent for networks which satisfy local balance. As in the figures, the reduced network of queue i becomes a closed tandem queue of queue i and its complement. The service of the complement queue is exponentially distributed with state dependent rate. So to queue i, it is equivalent a state dependent Poisson input with an arbitrary service time. It is a $M/G/1/\lambda_i$ queue. For service distribution with a rational Laplace transform, there is an embedded



Markov process. Then this embedded Markov process is solved as in the following figure.

Two parameters is checked to verify proper interface, mean queue length and throughputs of each node. Let q_i be the queue length of queue i and $E[q_i]$ be its mean value, t_i be the throughput(number of customers served per unit time) of queue i. The two conditions checked are:

$\sum_{i=1}^{M} E[q_i]=N$, the total number of customers in system.

$\sum_{i=1}^{M} t_i p_{ij}=t_j$, j=1,2,...,M, throughput into a node is equal to the throughput out of a node.

where p_{ij} is the probability a customer branches to queue j after service in queue i.

If the input process is Poissonian, then the service time of the pooled buffer queue is exponential distributed.

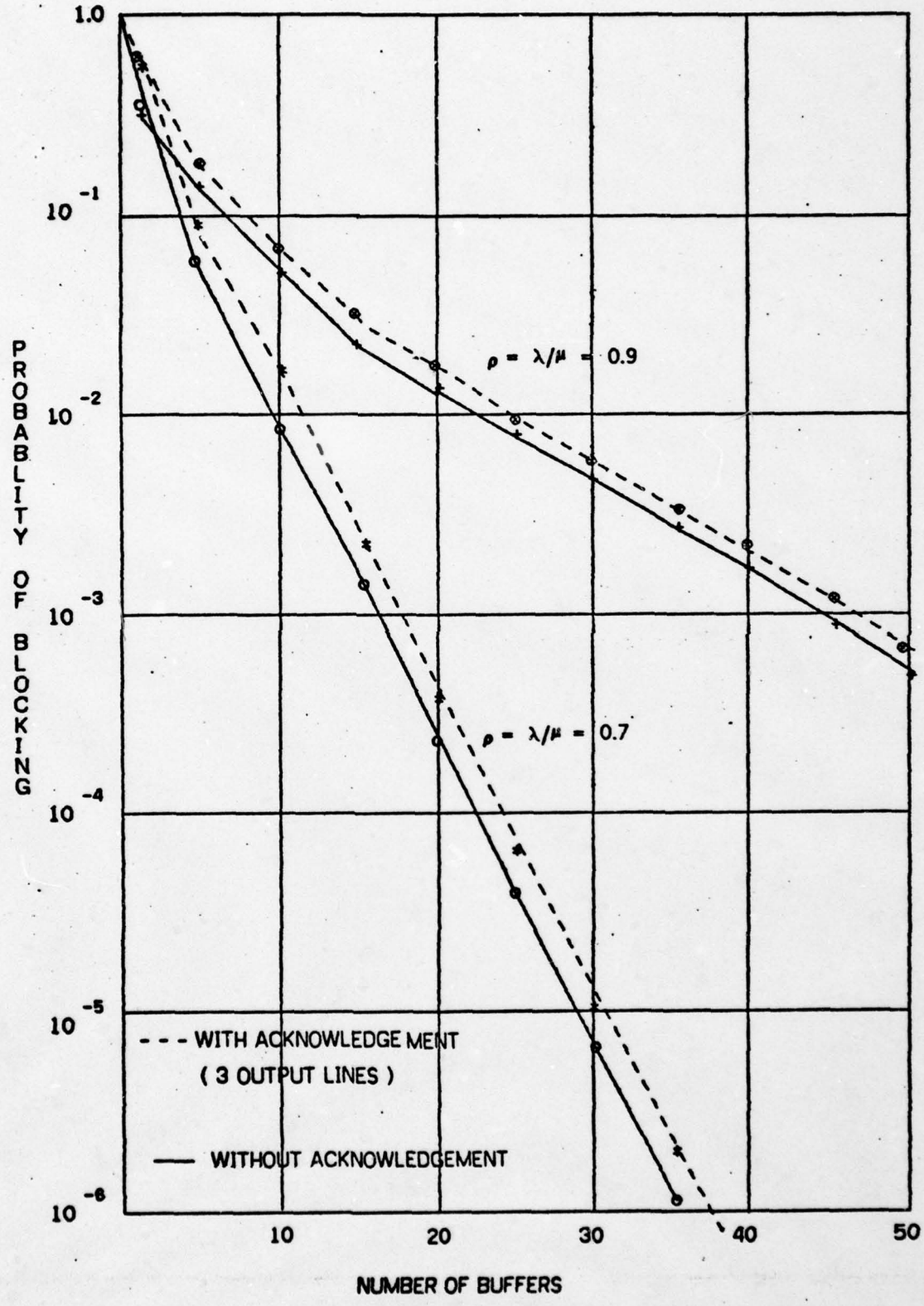

1.0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
PROBABLITY OF BLOCKING
$\rho = \lambda/\mu = 0.9$
$\rho = \lambda/\mu = 0.7$
- - - WITH ACKNOWLEDGE MENT
(3 OUTPUT LINES)
—— WITHOUT ACKNOWLEDGEMENT
10
20
30
40
50
NUMBER OF BUFFERS

FIG. 5

References

[ADC75] Advanced Data Communication Control Procedures (ADCCP), Independent Numbering. Proposed American National Standard. Fourth Draft, August 1975.

[Bar76] M.B.Barbacci and J.D.Oakley: The Integration of Circuit and Packet Switching Networks: Toward a SENNET Implementation. The 15th NBS-ACM Annual Technique Symposium, 1976.

[Bas75] F.Baskett et al.:Open, Close, and Mixed Networks of Queues with Different Class of Customers. JACM Vol.22, 1975

[Bha75] U.N.Bhat and M.J.Fischer: Multichannel Queueing System with Heterogeneous Classes of Arrivals. Defence Communication Engineering Center CP-75010, 1975.

[Buz73] .P.Buzen: Computational Algorithm for Closed Queueing Network with Exponential Servers. CACM Vol.16, 1973.

[CMU75] Carnegie-Mellon University: The Application of Multiple Processor Computer Systems to Digital Communications Networks. CMU Proposal No. 08103A to Defense Communications Engineering Center, Defense Communications Agency, May. 1975.

[Cha75A K.M.Chandy, U.Herzog, and L.Woo: Parametric Analysis of Queueing Networks. IBM. J. Res. Develop. 1975.

[Cha75B] K.M.Chandy, U.Herzog, and L.Woo: Approximation Analysis of General Queueing Network. IBM. J. Res. Develop. 1975.

[Cha72] J.H.Chang: Some Design and Evaluation Techniques of Data Communication Systems. IBM. Research Report RC-3736, 1972.

[Chu72] W.W.Chu and A.G.Konhein: On the Analysis and Modeling of a Class of Computer Communication Systems. IBM. Res. Report RC-3727, 1972.

[Cov75] Coviello G.J. and P.A. Vena: Integration of Circuit/Packet Switching in a SENET (Slotted Envelope NETwork) Concept. National Telecommunications Conference (NTC), New Orleans, December 1975.

[Fis75] M.J.Fischer and T.C.Harris: An Analysis of an Integrated Circuit and Packet Switched Telecommunication System. Defense Communication Engineering Center, TN 6-75, 1975.

[For75] Forgie, J.W., Speech Transmission in Packet Switched Store-and-Forward Networks. Proceedings of the National Computer Conference, NCC75, pp. 137:142.

[Hea73] Heart, F.E., S.M. Ornstein, W.R. Crowther, and W.B. Barker: A New Minicomputer/Multiprocessor for the ARPA Network. Proceedings. AFIPS NCC 42, 1973, pp. 529-537.

[Her75] U.Herzog, L.Woo and K.M.Chandy: Solution of Queueing Problems by a Recursive Technique. IBM. J. Res. Develop. 1975.

[Hou70] Hough, R.W., Future Data Traffic Volume, Institute of Electrical and Electronic Engineers, COMPUTER, Vol. 3, No. 5, September/October 1970, pp. 6-12.

[Jac63] J.R.Jackson: Jobshop-like Queueing System. Management Sci. Vol.10, 1963.

[Jon74] W.B.Jones and W.J.Thron: Numerical Stability in Evaluating Continued Fractions. Mathematics of Computation Vol.28, 1974.

[Kim75] Kimbleton, S.R. and G.M. Schneider: Computer Communication Networks: Approaches, Objectives, and Performance Considerations. ACM Computing Surveys, Vol. 7, No. 3, September 1975, pp. 129:173.

[Kle72] L.Kleinrock: Communication Nets: Stochastic Message Flow and Delay. Dover Publication, Inc. New York, 1972.

[Kuc72] A.Kuczura: Queues with Mixed Renewal and Poisson Input. Bell System Tech. Journal, Vol.51, 1972.

[Kum74] K.Kummerle: Multiplexor Performance for Integrated line and Packet-Switch Traffic. The Second International Coference on Computer Communication 1974.

[Lyo74] Lyons R.E.: Computer Communications in the Department of Defense. 13th Annual Technical Symposium, ACM Washington D.C. Chapter, Gaithersburg, Md. June 1974.

[Mei71] B.Meister, H.Muller and H.Rudin: Optimization of a New Model for message-switch Network. Proc. of Internation Conference on Communication 1971.

[Mit68] Mitrany,I. and Avi-Itzhak: A many-server queue with Service Interruptions. Operations Res. Vol.16,1968,pp.628.

[New75] Newell, A. and G. Robertson: Some Issues in Programming Multi-mini Processors. Department of Computer Science, Carnegie-Mellon University, June 1975.

[Red76] Reddy, D.R.: Speech Recognition by Machine: A Review. IEEE Proceedings, April 1976.

[Rob70] Roberts, L.G. and B.D. Wessler: Computer Network Development

to Achieve Resource Sharing SJCC Proceedings, Vol 36, 1970, pp. 543-549.

[Ros75] R.D.Rosner, R.H.Bittel and D.E.Brown: A High Throughput Packet-Switched Network Techniques without Message Reassembly. IEEE COM-23, 1975.

[Sch67] L.E.Schrage: The Queue M/G/1 with Feedback to lower priority Queues. Management Sci. Vol.13, 1967.

[Sie75] Siewiorek, D.P. and M.R. Barbacci: Modularity and Multiprocessor Structures: Some Open Problems in the Construction and Utilization of Mini- and Micro-Processor Networks. To appear in the Infotech State of the Art Report on Distributed Systems.

[Ver74] P.K.Verma and A.M.Rybczynski: The Economics of Segregated and Integrated Systems in Data Communication with Geometrically Distributed Message length. IEEE COM-22, 1974.

[Wul72] Wulf, W.A. and C.G. Bell: C.mmp: A Multi-Mini-Processor. Proceedings of the FJCC, 1972.

[Wul75] Wulf, W.A., R. Levin, and C. Pierson: An Overview of the HYDRA Operating System Development. Proceedings of the 5th ACM Symposium on Operating Systems Principles. Austin, Texas, November 1975.

Chapter 3

Reliability Evaluation : Alternative Architectures

TABLE OF CONTENTS

3.1. Reliability Considerations

3.1.1. Introduction

As the basic components of digital systems become more and more reliable over the years, systems sizes tend to grow larger and larger. Increasing also is the packaging density in an integrated circuit chip. This leads to increase in complexity of systems. In general, the more complex a system the less its reliability. Thus there seems to be an effect of diminising returns. Accurate reliability modeling is vital to correct prediction of reliability.

It is a common practice in reliability modeling to divide a system under investigation into a number of subsystems or modules. A judicious partitioning leads to a set of modules that are statistically mutually indenpedent. The reliability of the system is then merely the product of reliabilities of various modules.

The problem that still remains is that of finding the reliability of individual modules. In the days of discrete components it was a fairly well accepted practice to assume the statistical independence at the gate level, and raise the gate reliability to the number of gates in the module. Given the gate reliability, this yielded a good estimate for the module reliability. The present day technology of large scale integration renders the technique obsolete. Although still a function of the number of gates, the complexity may no longer be treated as a linear function of the number of gates. The following section outlines the currently acceptable approach.

3.1.2. Parts Count Model

Before getting to the parts count model, let us review the basic assumptions. It will be assumed the system is constructed of printed circuit boards. The PC boards hold IC chips. It is assumed that the system may be partitioned into chips and that the IC chips are statistically independent. It is further assumed that the reliability of a single module is exponentially distributed or that the failures of a single chip follow Poisson distribution. In other words,

$$\text{Probability of } k \text{ failures in time interval } (0,t) = e^{-\lambda t} (\lambda t)^k / (k!)$$

$$\text{Reliability} = \text{probability of no failures in } (0,t) = e^{-\lambda t}$$

With these assumptions, if a system does not contain any redundancy (i.e. every subsystem must function properly for the system to work), the system reliability is also exponential in nature. Furthermore, the failure rate of the system is the sum of failure rates of individual modules.

To estimate the failures rates of chips we will use the data published in Military Standardization Handbook 217-B [Mil74]. The handbook suggests the following model for the failure rate of a single chip.

$$\lambda = \pi_1 \pi_2 (\pi_3 d_1 G^{\alpha} + \pi_4 d_2 G^{\beta})$$

where d_1, d_2, α, β and π's are various constants,
G is the number of gates in the chip.

The π's are constants that depend on various factors, such as environment (whether the system is ground based, fixed or spaceborne, etc.), expected junction temperature, quality control (how rigorously has the chip been subjected to tests and burn-in). Table 3.1 shows the failure rates for chips with various number of gates. The system is assumed to be ground based, fixed, the quality control factor is assumed to

be 10 (which is in between the MIL-STD factor of 1.0 and the factor to be used for minimal quality control, 150), and the junction temperature is assumed to be 50. The other constants, d_1, d_2, α and β, are fixed and equal to 0.00129, 0.00389, 0.67 and 0.35 respectively.

Table 3.1

Gates	Failures in 10^6 hours	Gates	Failures in 10^6 hours
1	0.04342	9	0.10559
2	0.05711	10	0.11037
3	0.06721	11	0.11490
4	0.07553	12	0.11920
5	0.08275	13	0.12332
6	0.08920	14	0.12728
7	0.09508	15	0.13108
8	0.10052	16	0.13475

The estimation of failure rate for a module is best described by an example.

3.1.3. Example

As an example let us consider the Processor Interface module in C.mmp. Figure 3.1 shows the chip lay-out and list of parts for the Processor Interface. From this data we form the Table 3.2.

Table 3.2 Calculation of λ for Processor Interface

No.	Id	Gates	λ
1	74S140	4	0.07553
1	7440	4	0.07553
1	7404	6	0.08920
2	74S138	16	0.13475
6	7438	4	0.07553
13	74S74	10	0.11037

From individual λs in the table, we estimate λ for the Processor Interface to be

$$\lambda = 0.07553 + 0.07553 + 0.08920 + 2(0.13475) + 6(0.07553) + 13(0.11037)$$

$$= 2.39775 \text{ failures}/10^6 \text{ hours.}$$

Thus we arrive at the failure rate for Processor Interface. Similar calculations are carried out for all subsystems of C.mmp to yield the overall failure rate. The following table shows failure rates for the various subsystems of C.mmp. The processor-memory switch is seen to have the largest failure rate.

Table 3.3 Failure rates, C.mmp

Subsystem	λ in failures per million hours
11/40 Processor	57.5
Processor associated ckts.	11.41
Memory associated ckts.	7.14
Processor-memory switch	202.403
16K-words memory (core)	34.0

The following section will compare reliabilities of both C.mmp and CM* (Computer Modules) in both non-redundant and redundant configurations.

3.2. Reliability Comparison of C.mmp and Cm*

Parts count reliability models were developed for both of the in-house systems at CMU, the multiminiprocessor C.mmp and the computer modules system Cm*. As the conceptual diagrams of Figure 3.2 depict, both are general purpose multiprocessor systems. C.mmp is a general purpose system with a fixed architecture. Cm*, on the other hand, has a flexible architecture that may be so modified as to afford optimal performance for a given application.

As multiprocessor systems, both C.mmp and Cm* offer potential processing power well beyond that of a single processor. C.mmp has an upper limit of 16 processors (since the existing switch has only 16 ports for processors). In concept, the Cm* architecture is arbitrarily extendible; the only limiting factors are the cost and fundamental limits of algorithms programmed. The immediate goal of the Cm* project is to have a single-cluster 8-processor system functional by January 1977. When all of the processing power is not required, it is possible to conceive of either C.mmp or Cm* as a potentially redundant architecture. If a task requires the processing capability of, for example, only 4 processors, then we may view the other processors as stand-by spares. Assuming that we can detect and locate a faulty component (processor, memory, switch), and the malfunction was not an irrecoverable one, we can then logically replace a faulty component with a stand-by spare. The structures are thus fault-tolerant and have greater reliability.

To arrive at the reliabilities of multiprocessor fault-tolerant systems, we need to use two levels of modeling. We apply the parts count reliability model to estimate the failure rates of individual components. Then using the reliabilities of non-redundant components, we model the fault-tolerant system to arrive at a system reliability.

1. Parts Count Reliability Model : The failure rates for standard IC chips are found in Military Standardization Handbook (MIL-STD-HDBK-217B). Assuming exponential distribution for reliability and mutual statistical independence, the failure rate of each component is estimated as the sum of the failure rates of its various subcomponents. Since the handbook also predicts the failure rates for such subcomponents as resistors or printed boards, completeness of the model is assured. The following are the failure rates for various components of C.mmp and Cm* systems using the parts count reliability model.

	Component	Failure rate (failures per 10^6 hrs.)
C.mmp	PDP-11/40	57.496
	Processor associated circuitry (RELOC, processor interface)	11.414
	Memory box (16K words; core)	54.225
	Memory associated circuitry (Priority decode, etc.) per port	7.14
	Switch	202.403
Cm*	LSI-11 processor	109.0
	Memory (12K words; semiconductor)	203.343
	K.map	178.414

2. Redundancy Model : In the absence of realistic data on detection and replacement capabilities in multiprocessor systems, we use the following simplistic model (giving us the upper bound on potential reliability). If there are N identical components with the reliability of each component R_o, ($R_o = e^{-\lambda t}$ where λ = failure rate), and if a task requires k components, the subsystem can tolerate upto N-k failures, and the reliability of such a subsystem is

$$\sum_{i=0}^{N-k} \binom{N}{i} R_o^{N-i} (1 - R_o)^i$$

Thus the reliability of C.mmp with 16 processors and 16 64K-memories, with at least 4 processors and 4 memory ports required for the task, is

$$R_s \left(\sum_{i=0}^{12} \binom{16}{i} R_p^{16-i} (1 - R_p)^i \right) \left(\sum_{i=0}^{12} \binom{16}{i} R_m^{16-i} (1 - R_m)^i \right)$$

where R_s = switch reliability
R_p = (processor + associated circuitry) reliability
R_m = (memory + associated circuitry) reliability.

A similar approach is applicable to Cm* system. The reliability of a single cluster 8-processor Cm*, for a task requiring 4 processors and 4 memories, is

$$R_{km} \left(\sum_{i=0}^{8} \binom{8}{i} R_{cm}^{8-i} (1 - R_{cm})^i \right)$$

where R_{km} = reliability of (K.map + map bus)
R_{cm} = reliability of a computer module
= product of reliabilities of S.local, memory and processor.

The reliabilities of the two redundant systems described by the two equations above are calculated and are presented in Figures 3.3 and 3.4. Figure 3.3 shows the reliabilities of a 16-processor C.mmp as a function of time for tasks requiring various number of processors. The plot for task processor = 16 is the reliability of a totally non-redundant C.mmp. The dramatic increase in reliability as the number of task processors decreases is evident. Figure 3.4 depicts similar plots for a single cluster Cm* system. Again there is noticeable improvement in reliability as we move from a non-redundant to redundant system.

To compare the two systems, we use the notion of mission time improvement (MTI) [Kno64]. If the minimum reliability desired is R_{sysmin}, the mission time tm of a system is defined by $R_{sys}(tm) = R_{sysmin}$. The MTI of a system 1 over a system 2 is then defined to be tm_1/tm_2. It is the measure of how much longer the system 1 is able to keep its reliability above the minimum required than the system 2. To compare the two systems C.mmp and Cm*, we will use LSI-11 as a base. We find the MTI of C.mmp over LSI-11 by t_1/t_2 where $R_{c.mmp}(t_1) = R_{LSI-11}(t_2)$. Since the MTI is not constant over all values of R_{sysmin}, we plot these as a function of R_{LSI-11} in Figure 3.5. A similar graph

is plotted for MTI of Cm* over LSI-11 in Figure 3.6. In order to keep the comparison fair, the memory boxes for C.mmp were normalized to 16K words in Figure 3.5.

The plots of MTI show the 16-processor C.mmp to be slightly less reliable than a single cluster 8-processor Cm*. The chief reason for this is that although C.mmp has more processors and allows greater redundancy, the non-redundant part, the switch, is considerably less reliable than the corresponding Cm* component, K.map. When Cm* achieves its more distant goals of having a multi-cluster system, normalization of processing power, memory capacity, and redundancy will be required.

Up to this point, we have been discussing the reliabilities of both non-redundant and redundant systems in a long mission mode. No repairs were presumed. However, in systems such as the integrated communications switch, periodic maintenance is not only possible but is also highly advisable. The following treatment discusses the effect of maintenance on the system reliability.

3.3. Effect of Periodic Maintenance on Reliability

3.3.1. Introduction

In an attempt to increase the life of a non-perfect system, various redundancy techniques have been applied. It has been shown that TMR (triple modular redundancy) with stand-by sparing can be used to achieve improvement in reliability and expected life. While the general analysis holds perfectly well for systems performing vital functions in a space mission, it is extremely pessimistic in a more commonly used system where a technician may be able to perform repairs. Even more common is a situation in which certain tests are applied to the system at regular intervals to insure its integrity, followed by any equired repair. It is to be expected in most cases that the life span of a system subjected to such maintenance would be greater than that of an identical system left unattended. In the following analysis we estimate the expected life of a non-perfect system under periodic maintenance.

3.3.2. Life of An Unmaintained System

As has been the common practice, we will assume the failures in a non-redundant system to have an exponential distribution. We will denote the failure rate by λ. The reliability of the system (i.e. the probability that there is no failure during the time interval (0,t)) is then $R(t) = e^{-\lambda t}$. The life of the system is estimated as follows.

Let T be the time at which a failure occurs. Then the distribution function F (t) is

F (t) = Prob (the failure occuring in (0,t))

$= \text{Prob}(0<T<t)$

$= 1 - \text{Prob}(T>t)$

$= 1 - R(t)$

The life of the system is the expected value of T, $E(T)$.

And, $E(T) = \int_0^\infty (1 - F(t))\, dt$

$= \int_0^\infty R(t)\, dt$

Note that the above equation is a general one, and may be applied to any function R(t). Also note that according to the equation above, $E(T)$, or the life of the system, is the same as the area under the curve for any function R(t). For the exponential distribution,

$$E(T) = 1/\lambda.$$

3.3.3. Life of A Maintained System

Let us now consider a system being operated under periodic maintenance. We assume that certain tests are performed at a regular time interval, β. Every occasion on which these tests are performed with success (or the failures of the tests are followed by subsequent necessary repairs), there is a lesser probability of failure in immediate future. This leads to an increase in reliability after every maintenance period. We shall consider two models for the improvement obtained by periodic maintenance.

Model I : Improvement through maintenance = $d\,(1 - e^{-\lambda\beta})$, where d is a constant, $0 \le d \le 1$. This model assumes that a constant fraction of the lost reliability is recovered. Figure 3.7 shows a hypothetical reliability function using this assumption.

Let $R_i(t)$ = reliability function during the ith interval.

Then,

$$R_1(t) = e^{-\lambda t}$$

$$R_2(t) = \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}\, e^{-\lambda t}$$

$$R_3(t) = \{ e^{-2\lambda\beta} + d\,(1 - e^{-\lambda\beta})\, e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}\, e^{-\lambda t}$$

$$= \{ e^{-2\lambda\beta} + d\,(1 - e^{-2\lambda\beta}) \}\, e^{-\lambda t}$$

in general,

$$R_{i+1}(t) = \{ e^{-i\lambda\beta} + d\,(1 - e^{-i\lambda\beta}) \}\, e^{-\lambda t}$$

The area under the (i+1)st segment,

$$A_{i+1} = \int_0^{\beta} R_{i+1}(t)\, dt$$

$$= \{ e^{-i\lambda\beta} + d\,(1 - e^{-i\lambda\beta}) \} \int_0^{\beta} e^{-\lambda t}\, dt$$

$$= (1/\lambda)\,(1 - e^{-\lambda\beta})\, \{ e^{-i\lambda\beta} + d\,(1 - e^{-i\lambda\beta}) \}$$

The attempt to evaluate life, which is also the sum of all A_i's, yields ∞ for an answer. This is due to the fact that under the assumptions, the reliability function reaches a steady state as shown in Figure 3.8. The area under this reliability function is not finite.

Since the tests will not, in general, test all possible system components, there will remain components that are never replaced or repaired during the periodic maintenance. These components will eventually fail. Thus an expected infinite life span is unrealistic.

Model II : In the first model, we assumed the improvement due to maintenance to be a constant. In a more pessimistic model, we may assume that the improvement in reliability due to maintenance at the end of the ith period is a fraction of the reliability lost in the ith period. In other words,

$$R_{i+1}(t) = \{ R_{io}\, e^{-\lambda\beta} + d\, R_{io}\, (1 - e^{-\lambda\beta}) \}\, e^{-\lambda t}$$

where $R_{io} = R_i$ at beginning of ith period.

Writing R_i's explicitly,

$$R_1(t) = e^{-\lambda t}$$

$$R_2(t) = \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}\, e^{-\lambda t}$$

$$R_3(t) = [\, \{e^{-\lambda\beta} + d(1-e^{-\lambda\beta})\}e^{-\lambda\beta} + d\{e^{-\lambda\beta} + d(1-e^{-\lambda\beta})\}(1-e^{-\lambda\beta}) \,]\, e^{-\lambda t}$$

$$= \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}^2\, e^{-\lambda t}$$

and

$$R_{i+1}(t) = \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}^i\, e^{-\lambda t}$$

The area under the ith segment,

$$A_{i+1} = \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}^i \int_0^{\beta} e^{-\lambda t}\, dt$$

$$= (1/\lambda)\,(1 - e^{-\lambda\beta})\, \{ e^{-\lambda\beta} + d\,(1 - e^{-\lambda\beta}) \}^i$$

And now the expected life is given by

$$\text{life} = \sum A_{i+1}$$

$$= (1/\lambda)\,(1 - e^{-\lambda\beta})\, \{ 1 - e^{-\lambda\beta} - d\,(1 - e^{-\lambda\beta}) \}^{-1}$$

$$= 1 \,/\, \{ \lambda\,(1 - d) \}$$

The life span is thus improved by a factor of $(1 - d)^{-1}$. If $d = 1$, the maintenance is perfect and the life span tends to be infinite. If $d = 0$, we revert back to an unmaintained system with life span $(1/\lambda)$.

3.3.4. Redundant Systems With Maintenance

Until now we have considered a non-redundant system with $R(t) = e^{-\lambda t}$. Where reliabilities of high order (better than 1 failure in a million hours) are required, improvements through technological advances alone fall short of the objective. System designers have resorted to redundancy to attain higher reliabilities. Triple modular redundancy (TMR) with majority voting is one of the redundancy techniques used. Since such a technique allows the system to tolerate a single failure in any of the three modules, the reliability of a TMR system is

$$R_{sys}(t) = R^3(t) + 3\,R^2(t)\,(1 - R(t))$$

$$= 3\,R^2(t) - 2\,R^3(t)$$

where R(t) = reliability of non-redundant system.

Since we intend to keep the development applicable in general, we will use $R_{sys}(t)$ during the discussion and substitute the expression for a TMR system only to exemplify the results.

3.3.5. Life of An Unmaintained, Redundant System

Recalling that the life span of a system, E(*T*), is the same as that of the area under the reliability function we can write

$$\text{Life} = \int_0^\infty R_{sys}(t)\,dt$$

For a TMR system,

$$\text{Life (TMR)} = \int_0^\infty \{\, 3\,R^2(t) - 2\,R^3(t) \,\}\,dt$$

$$= (1/\lambda)\,(\,3/2 - 2/3\,)$$

$$= 5\,/\,(6\,\lambda)$$

Again, we now focus our attention to a system under periodic maintenance.

3.3.6. Life of A Maintained, Redundant System

We will again consider the two models. Under the first simple model, we assume that the improvement through maintenance is a constant fraction of the probability of failure, i.e. d (1 - $R_{sys}(\beta)$) where β is the period of the maintenance cycle.

Let $R_i(t)$ = system reliability function during the ith interval.

Then,

$$R_i(t) = R_{sys}(t)$$

$$R_2(t) = \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta)\,\} R_{sys}(t)$$

$$R_3(t) = \{ R^2_{sys}(\beta) + d\,(1 - R_{sys}(\beta))\,R_{sys}(\beta) + d\,(1 - R_{sys}(\beta))\,\} R_{sys}(t)$$

$$= \{ R^2_{sys}(\beta) + d\,(1 - R^2_{sys}(\beta)\,\} R_{sys}(t)$$

And,

$$R_i(t) = \{ R^i_{sys}(\beta) + d\,(1 - R^i_{sys}(\beta)\,\} R_{sys}(t)$$

The area under the (i+1)st segment is

$$A_{i+1} = \int_0^\beta \{ R^i_{sys}(\beta) + d\,(1 - R^i_{sys}(\beta)\,\} R_{sys}(t)\,dt$$

$$= \{ R^i_{sys}(\beta) + d\,(1 - R^i_{sys}(\beta)\,\} \int_0^\beta R_{sys}(t)\,dt$$

For a TMR system,

$$R_{sys}(t) = 3\,e^{-2\lambda t} - 2\,e^{-3\lambda t}$$

and

$$\int_0^\beta R_{sys}(t)\,dt$$

$$= \int_0^\beta \{\, 3\,e^{-2\lambda t} - 2\,e^{-3\lambda t}\,\}\,dt$$

$$= (3 / 2\lambda)\,(1 - e^{-2\lambda\beta}) - (2 / 3\lambda)\,(1 - e^{-3\lambda\beta})$$

We again note that when we try to sum all A_i's, the results approaches .

Considering the second model that we suggested earlier, where the improvement in reliability is a fraction of the reliability lost in the ith period, we may write

$$R_{i+1}(t) = \{ R_i(\beta) + d\,R_i(0)\,(1 - R_{sys}(\beta))\,\} R_{sys}(t)$$

Writing R_i's explicitly, we have

$$R_1(t) = R_{sys}(t)$$

$$R_2(t) = \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta))\,\} R_{sys}(t)$$

$$R_3(t) = \{ R^2_{sys}(\beta) + d\,R_{sys}(\beta)\,(1 - R_{sys}(\beta)) + d\,R_{sys}(\beta) + d^2\,(1 - R_{sys}(\beta)$$

$$- d\,R_{sys}(\beta) - d^2\,R_{sys}(\beta)\,(1 - R_{sys}(\beta))\,\} R_{sys}(t)$$

$$= \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta))\,\}^2 R_{sys}(t)$$

And, in general,

$$R_{i+1}(t) = \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta)) \}^{i}\, R_{sys}(t)$$

The area under the (i+1)st segment,

$$A_{i+1} = \int_0^{\beta} R_{i+1}(t)\, dt$$

$$= \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta)) \}^{i} \int_0^{\beta} R_{sys}(t)\, dt$$

Then the life span for the system is

$$\text{Life} = \sum A_{i+1}$$

$$= \left(\int_0^{\beta} R_{sys}(t)\, dt \right) \sum \{ R_{sys}(\beta) + d\,(1 - R_{sys}(\beta)) \}^{i}$$

$$= \left(\int_0^{\beta} R_{sys}(t)\, dt \right) \{ 1 - R_{sys}(\beta) - d\,(1 - R_{sys}(\beta) \}^{-1}$$

$$= \left(\int_0^{\beta} R_{sys}(t)\, dt \right) (1 - R_{sys}(\beta))^{-1}\, (1 - d)^{-1}$$

For a TMR system,

$$\int_0^{\beta} R_{sys}(t)\, dt = (3 / 2\lambda)\,(1 - e^{-2\lambda\beta}) - (2 / 3\lambda)\,(1 - e^{-3\lambda\beta})$$

and $R_{sys}(\beta) = 3\, e^{-2\lambda\beta} - 2\, e^{-3\lambda\beta}$.

Substituting these two expressions in the equation for Life we can estimate the life span of a TMR system. As the expression is very complex, the improvement is not obvious in this form. Let us, therefore, consider numerical examples. The following table (Table 3.4) allows the comparison in the life spans of unmaintained and maintained TMR systems.

The data conform to the expectations. As the period maintenance becomes larger, the expected life becomes shorter. We note that after $\beta = 10000$, the life is shorter than the period of maintenance. Thus the maintenance has no effect on the performance for any period larger than 10000. The life depends more strongly on d, as seen by the sharp increase of life as d approaches unity. The constant d is a function of how efficient maintenance routines are. Since the expected life is extremely sensitive to the effectiveness of maintenance around d = 0.9, slight improvement in

Table 3.4 Expected life with periodic maintenance; Lamda = 0.000100

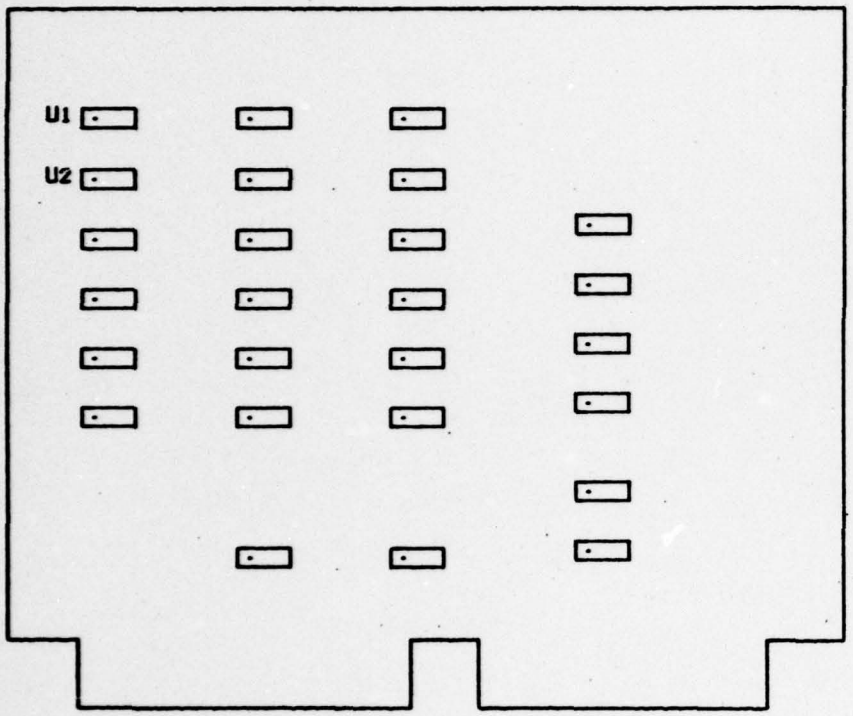
Life of an unmaintained TMR system : 8333.33

Life of a maintained system :

d	beta 10	100	1000	10000	100000	1000000
0.10	3709894.97	376550.34	43282.55	10629.76	9259.26	9259.26
0.20	4173631.84	423619.13	48692.87	11958.48	10416.67	10416.67
0.30	4769864.97	484136.15	55648.99	13666.83	11904.76	11904.76
0.40	5564842.45	564825.50	64923.83	15944.64	13888.89	13888.89
0.50	6677810.96	677790.60	77908.59	19133.57	16666.67	16666.67
0.60	8347263.73	847238.26	97385.74	23916.96	20833.33	20833.33
0.70	11129684.76	1129650.99	129847.65	31889.28	27777.78	27777.78
0.80	16694527.14	1694476.49	194771.48	47833.92	41666.67	41666.67
0.90	33389054.29	3388952.97	389542.96	95667.84	83333.33	83333.33
0.90	33389054.29	3388952.97	389542.96	95667.84	83333.33	83333.33
0.91	37098948.28	3765503.21	432825.50	106297.59	92592.59	92592.59
0.92	41736319.41	4236191.38	486928.72	119584.80	104166.67	104166.67
0.93	47698649.49	4841361.44	556489.95	136668.34	119047.62	119047.62
0.94	55648422.43	5648254.82	649238.25	159446.39	138888.88	138888.88
0.95	66778103.59	6777905.44	779085.86	191335.66	166666.65	166666.65
0.96	83472638.82	8472382.75	973857.43	239169.60	208333.34	208333.34
0.97	111296844.85	11296509.63	1298476.49	318892.78	277777.77	277777.77
0.98	166945246.55	16944762.34	1947714.50	478339.11	416666.60	416666.60
0.99	333890617.48	33889537.31	3895430.45	956678.58	833333.51	833333.51

maintenance routines may be rewarded with substantially greater life for the system. This fact is emphasized when we plot the expected life of a TMR system as a function in Figure 3.9.

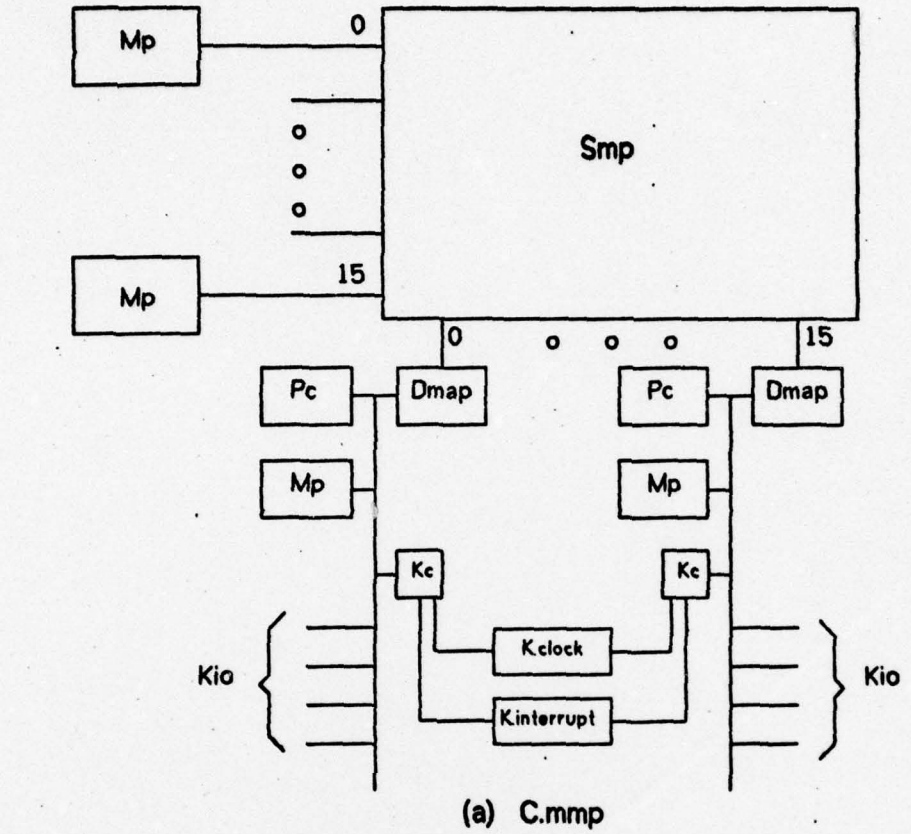
LAY OUT



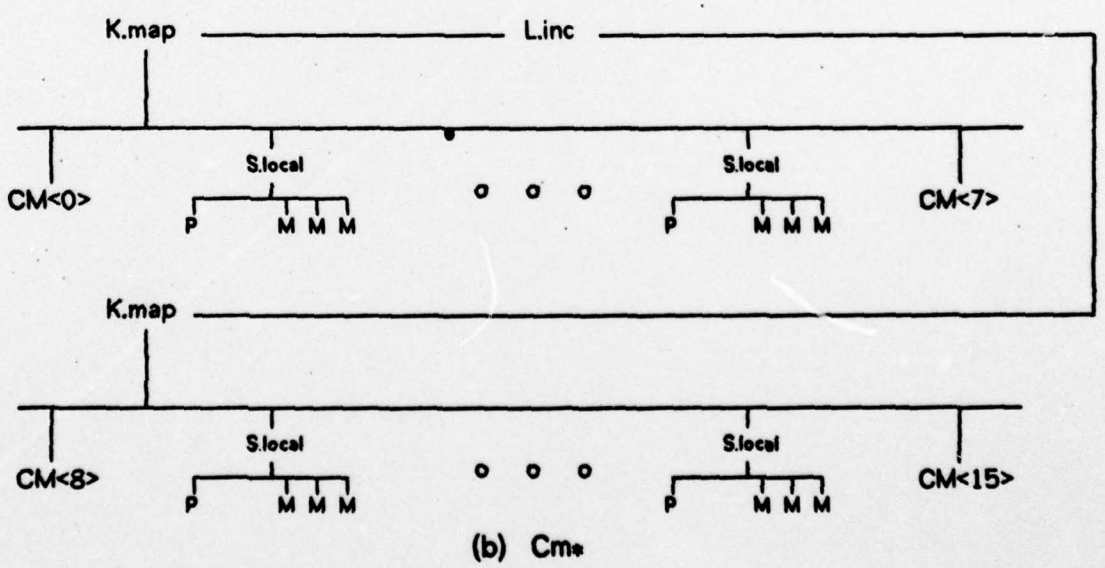
LIST OF PARTS

U1	SN74S74
U2	SN7438
U3	SN74S74
U4	SN74S74
U5	SN7438
U6	SN74S74
U7	SN74S74
U8	SN7438
U9	SN74S74
U10	SN74S74
U11	SN7438
U12	SN74S74
U13	SN74S74
U14	SN7438
U15	SN74S74
U16	SN74S74
U17	SN7438
U18	SN74S74
U19	SN74S140
U20	SN7440
U21	SN7404
U22	SN74S74
U23	SN74S138
U24	SN74S138

FIGURE 3.1 Processor Interface



(a) C.mmp



(b) Cm*

FIGURE 3.2 General structures of the two multiprocessor systems, C.mmp and Cm*

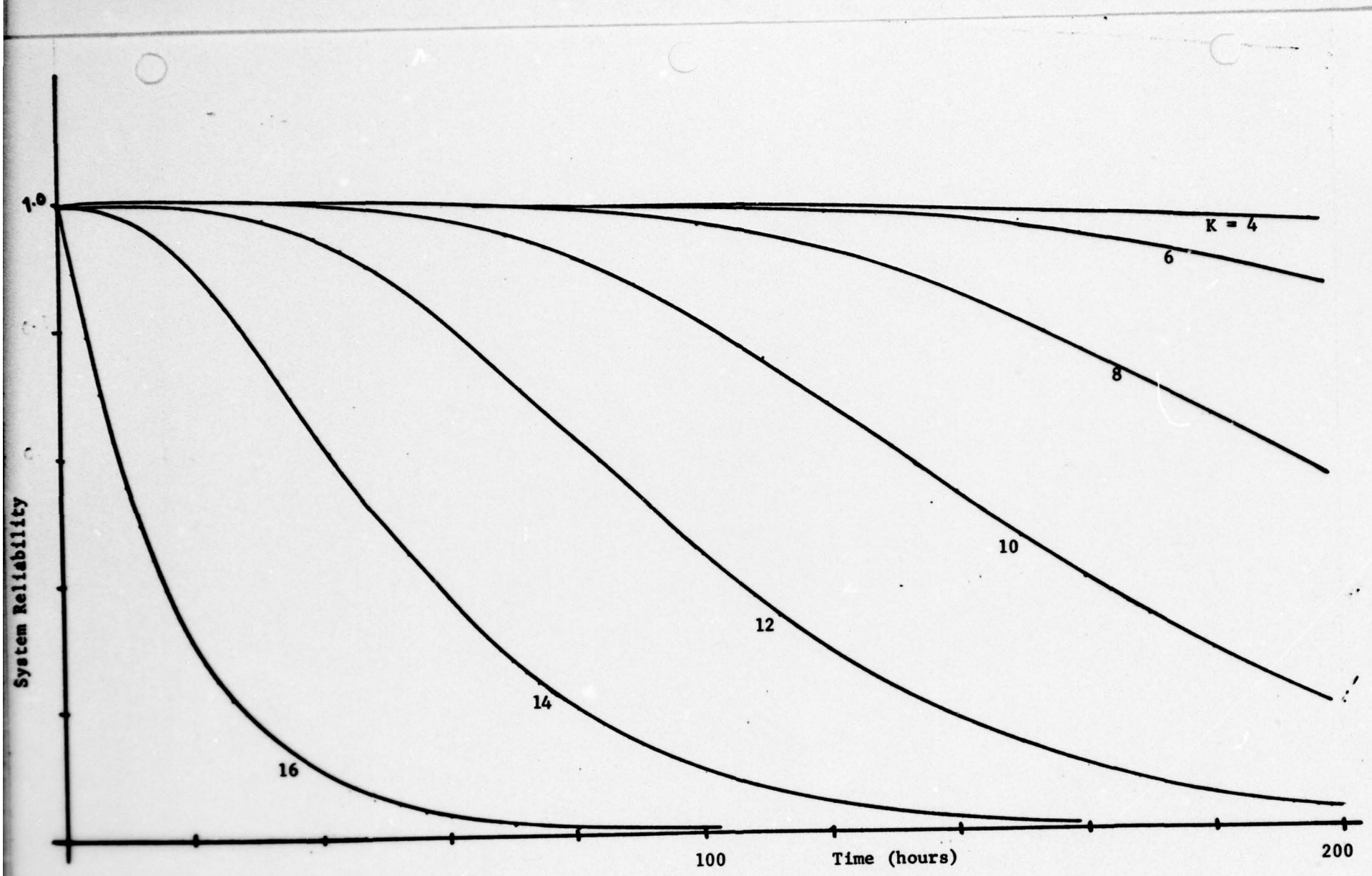


Fig. 3.3: Reliability of C.mmp for a task requiring K processors.

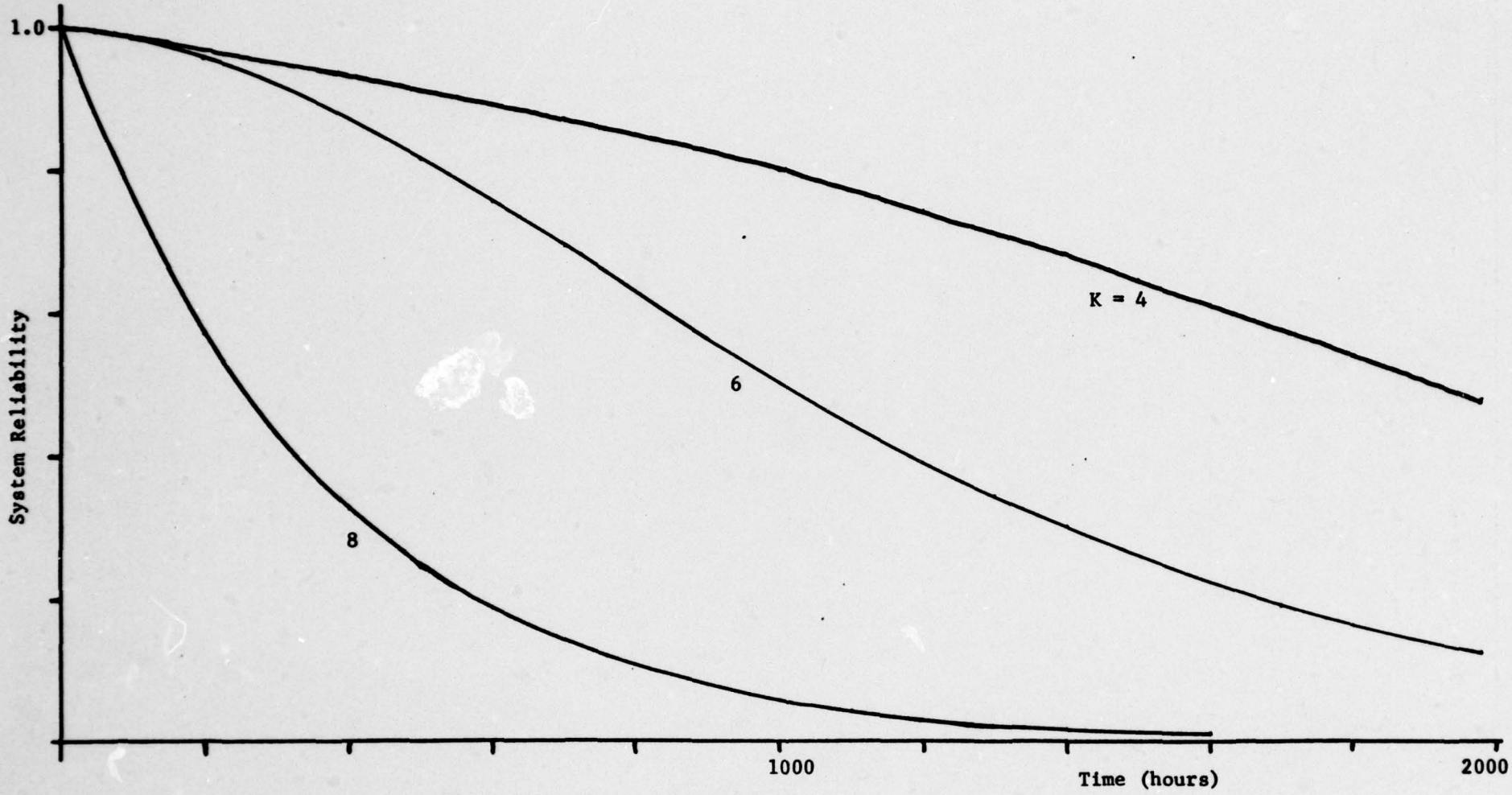


Fig. 3.4: Reliability of Cm* for a task requiring K processors.

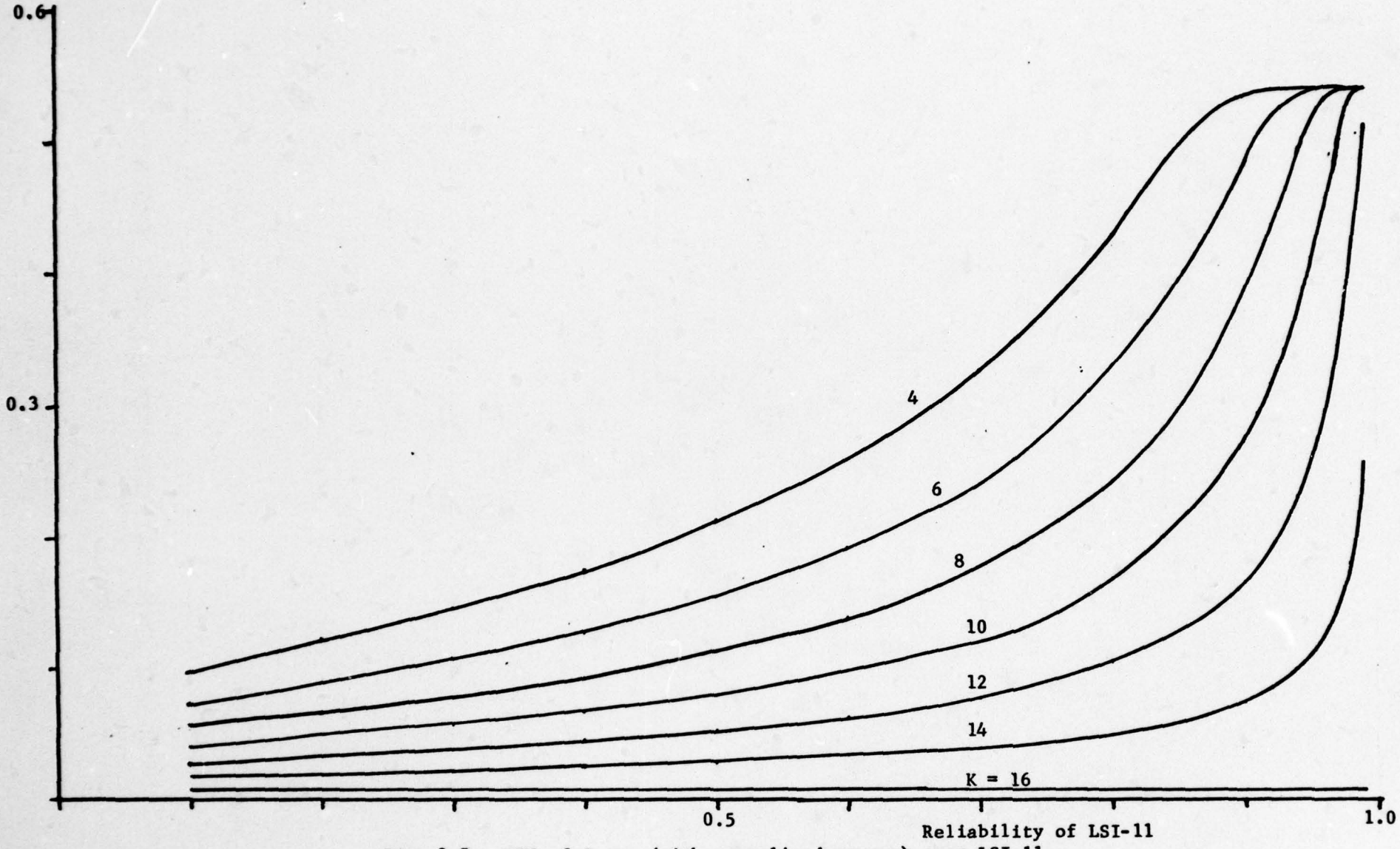


Figure 3.5: MTI of C.mmp (with normalized memory) over LSI-11.

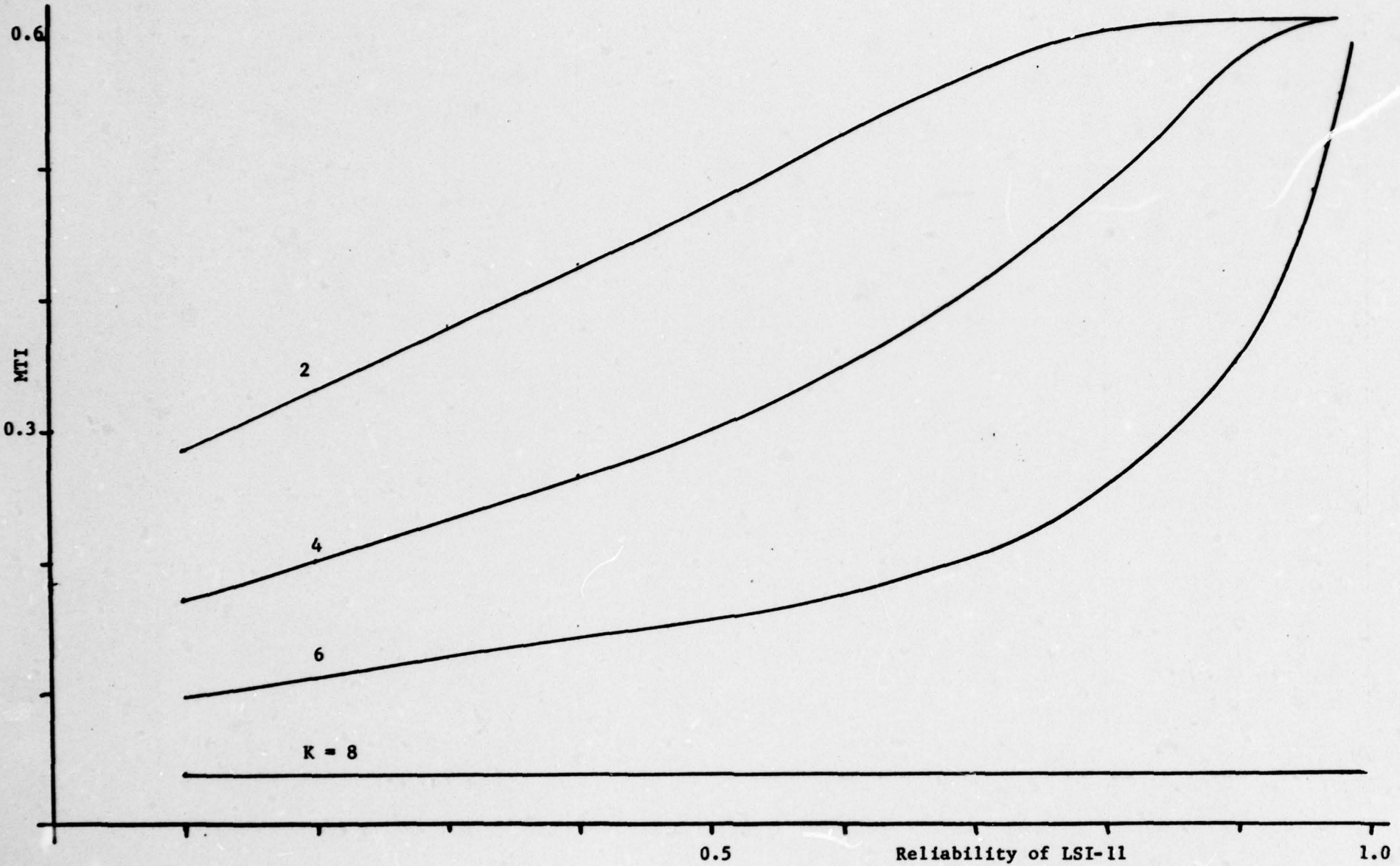


Figure 3.6: MTI of a single cluster Cm* over LSI-11.

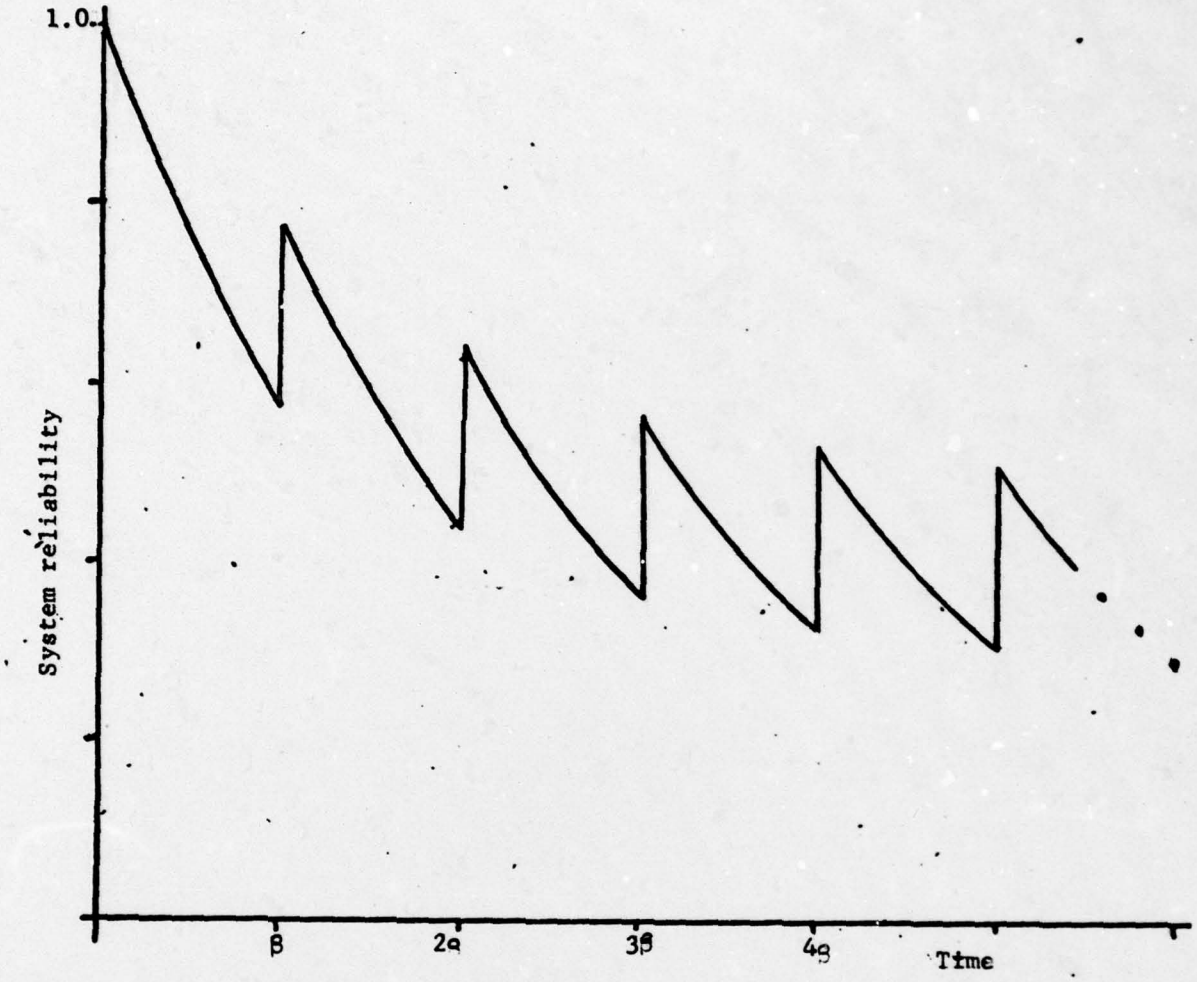


Figure 3.7 : Effect of periodic maintenance on system reliability.

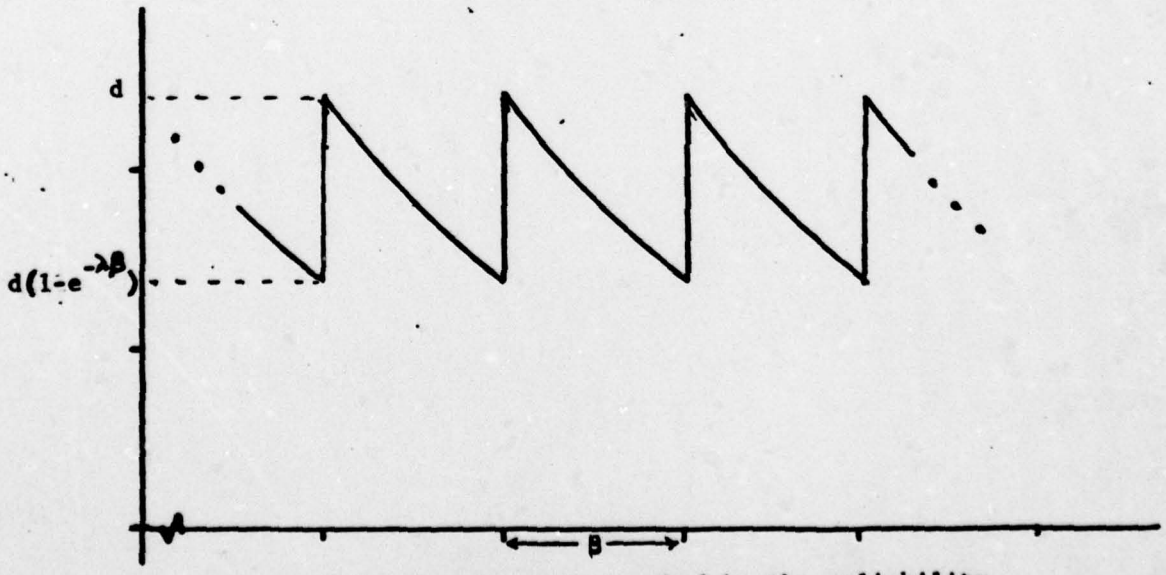
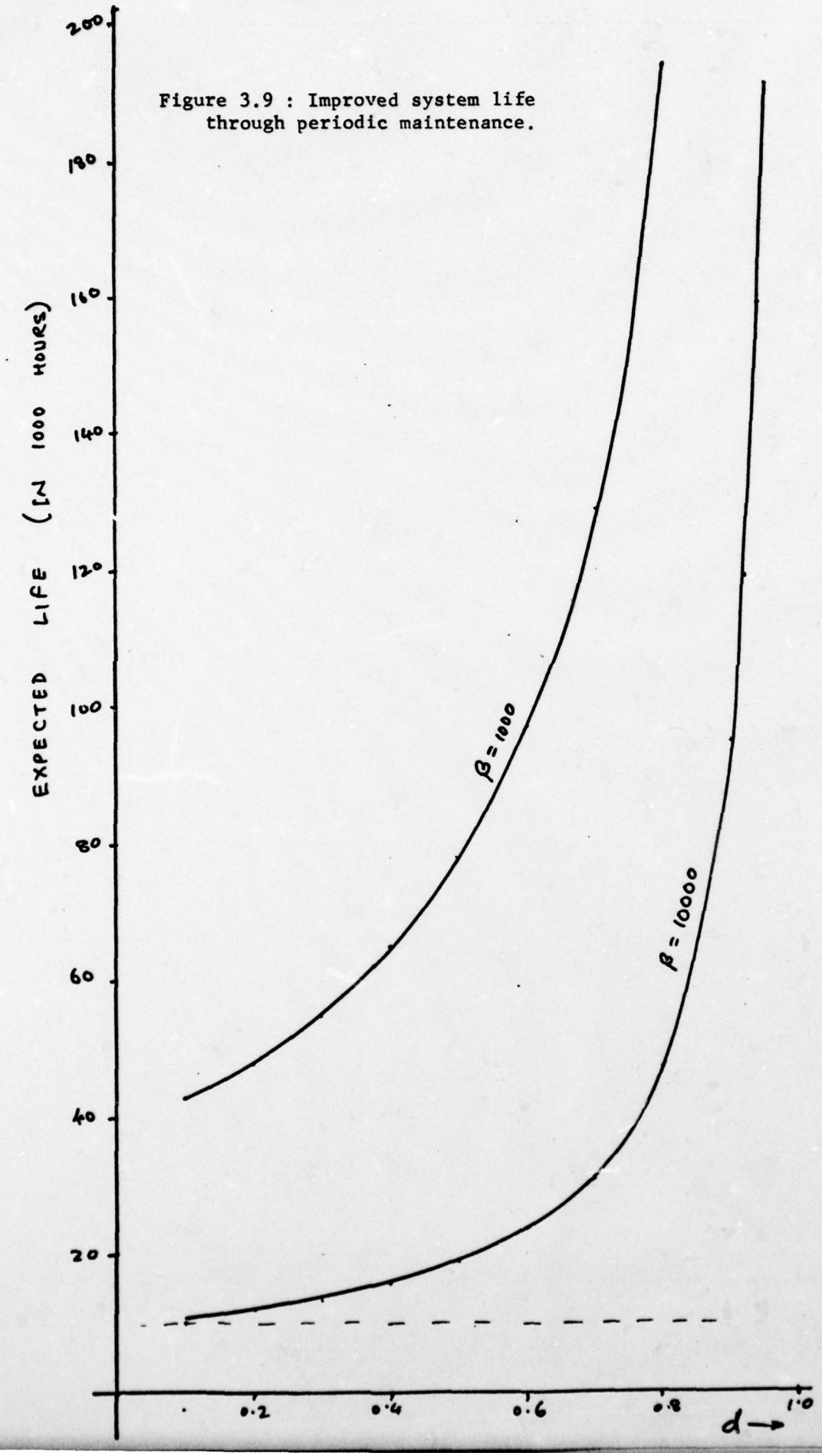


Figure 3.8 : Steady state reached by the reliability function in the first model for periodic maintenance.

Figure 3.9 : Improved system life through periodic maintenance.



References

[ADC75] Advanced Data Communication Control Procedures (ADCCP), Independent Numbering. Proposed American National Standard. Fourth Draft, August 1975.

[Bar76] M. B. Barbacci and J. D. Oakley: The Integration of Circuit and Packet Switching Networks: Toward a SENNET Implementation. The 15th NBS-ACM Annual Technique Symposium, 1976.

[Bas75] F. Baskett et al. : Open, Close, and Mixed Networks of Queues with Different Class of Customers. JACM Vol. 22, 1975

[Bha75] U. N. Bhat and M. J. Fischer: Multichannel Queueing System with Heterogeneous Classes of Arrivals. Defence Communication Engineering Center CP-75010 and TC 21-75.

[Buz73] P. Buzen: Computational Algorithm for Closed Queueing Network with Exponential Servers. CACM Vol. 16, 1973.

[CMU75] Carnegie-Mellon University: The Application of Multiple Processor Computer Systems to Digital Communications Networks. CMU Proposal No. 08103A to Defense Communications Engineering Center, Defense Communications Agency, May 1975.

[Cha75A] K. M. Chandy, U. Herzog, and L. Woo: Parametric Analysis of Queueing Networks. IBM. J. Res. Develop. 1975.

[Cha75B] K. M. Chandy, U. Herzog, and L. Woo: Approximation Analysis of General Queueing Network. IBM. J. Res. Develop. 1975.

[Cha72] J. H. Chang: Some Design and Evaluation Techniques of DAta Communication Systems. IBM. Research Report RC-3736, 1972.

[Chu72] W. W. Chu and A. G. Konhein: On the Analysis and Modeling of a Class of Computer Communication Systems. IBM. Res. Report RC-3727, 1972.

[Cov75] G. J. Coviello and P. A. Vena: Integration of Circuit/Packet Switching in a SENET (Slotted Envelope NETwork) Concept. National Telecommunications Conference (NTC), New Orleans, December 1975.

[Fis75] M. J. Fischer and T. C. Harris: An Analysis of an Integrated Circuit and Packet Switched Telecommunication System. Defense Communication Engineering Center, TN 6-75, 1975.

[For75] J. W. Forgie: Speech Transmission in Packet Switched Store-and-Forward Networks. Proceedings of the National Computer Conference, NCC75, pp. 137: 142.

[Hea73] *Heart, F. E. , S. M. Ornstein, W. R. Crowther, and W. B. Barker:* A New Minicomputer/Multiprocessor for the ARPA Network. Proceedings. AFIPS NCC 42, 1973, pp. 529-537.

[Her75] U. Herzog, L. Woo and K. M. Chandy: Solution of Queueing Problems by a Recursive Technique. IBM. J. Res. Develop. 1975.

[Hou70] R. W. Hough: Future Data Traffic Volume, Institute of Electrical and Electronic Engineers, COMPUTER, Vol. 3, No. 5, September/October 1970, pp. 6-12.

[Jac63] J. R. Jackson: Jobshop-like Queueing System. Management Sci. Vol. 10, 1963.

[Jon74] W. B. Jones and W. J. Thron: Numerical Stability in Evaluating Continued Fractions. Mathematics of Computation Vol. 28, 1974.

[Kim75] Kimbleton, S. R. and G. M. Schneider: Computer Communication Networks: Approaches, Objectives, and Performance Considerations. ACM Computing Surveys, Vol. 7, No. 3, September 1975, pp. 129: 173.

[Kle72] L. Kleinrock: Communication Nets: Stochastic Message Flow and Delay. Dover Publication, Inc. New York, 1972.

[Kno64] J. K. Knox-Smith: Improving the Reliability of Digital Systems by Redundancy and Restoring Organs, Solid-State Electronics Lab. , Technical Report No. 4186-2, Stanford University, Stanford, California, August 1964.

[Kuc72] A. Kuczura: Queues with Mixed Renewal and Poisson Input. Bell System Tech. Journal, Vol. 51, 1972.

[Kum74] K. Kummerle: Multiplexor Performance for Integrated line and Packet-Switch Traffic. The Second International Coference on Computer Communication 1974.

[Lyo74] R. E. Lyons: Computer Communications in the Department of Defense. 13th Annual Technical Symposium, ACM Washington D. C. Chapter, Gaithersburg, Md. June 1974.

[Mei71] B. Meister, H. Muller and H. Rudin: Optimization of a New Model for message-switch Network. Proc. of Internation Conference on Communication 1971.

[Mil74] Military Standardization Handbook: Reliability Prediction of Electronic Equipment, September 1974.

[New75] A. Newell, and G. Robertson: Some Issues in Programming Multi-mini Processors. Department of Computer Science, Carnegie-Mellon University, June 1975.

[Red76] D. R. Reddy: Speech Recognition by Machine: A Review. IEEE Proceedings, April 1976.

[Rob70] Roberts, L. G. and B. D. Wessler: Computer Network Development to Achieve Resource Sharing SJCC Proceedings, Vol 36, 1970, pp. 543-549.

[Ros75] R. D. Rosner, R. H. Bittel and D. E. Brown: A High Throughput Packet-Switched Network Techniques without Message Reassembly. IEEE COM-23, 1975.

[Sch67] L. E. Schrage: The Queue M/G/1 with Feedback to lower priority Queues. Management Sci. Vol. 13, 1967.

[Sie75] D. P. Siewiorek and M. R. Barbacci: Modularity and Multiprocessor Structures: Some Open Problems in the Construction and Utilization of Mini- and Micro-Processor Networks. To appear in the Infotech State of the Art Report on Distributed Systems.

[Ver74] P. K. Verma and A. M. Rybczynski: The Economics of SEgrated and Integrated Systems in Data Communication with Geometrically Distributed Message length. IEEE COM-22, 1974.

[Wul72] W. A. Wulf and C. G. Bell: C. mmp: A Multi-Mini-Processor. Proceedings of the FJCC, 1972.

[Wul75] W. A. Wulf, R. Levin, and C. Pierson: An Overview of the HYDRA Operating System Development. Proceedings of the 5th ACM Symposium on Operating Systems Principles. Austin, Texas, November 1975.